Offline data processing software for the High Energy cosmic-Radiation Detection facility (HERD)

Reporter： Qianqian Shi

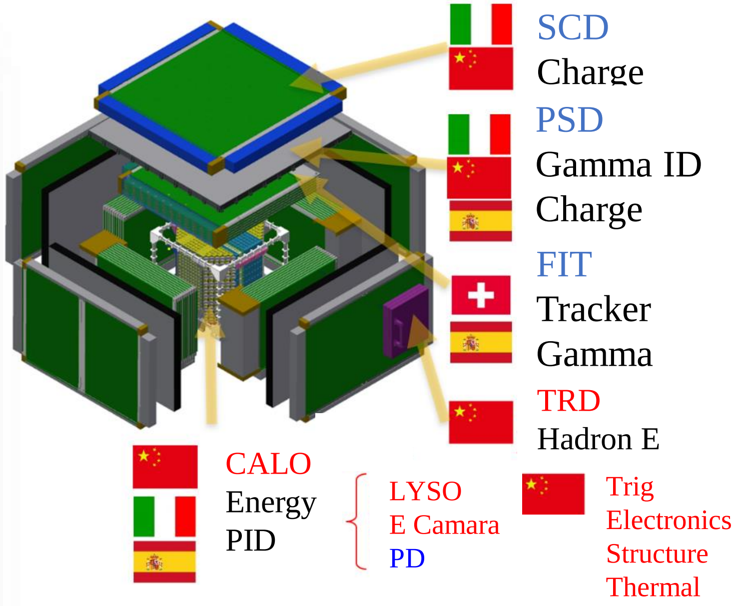
On behalf of the HERD offline software team

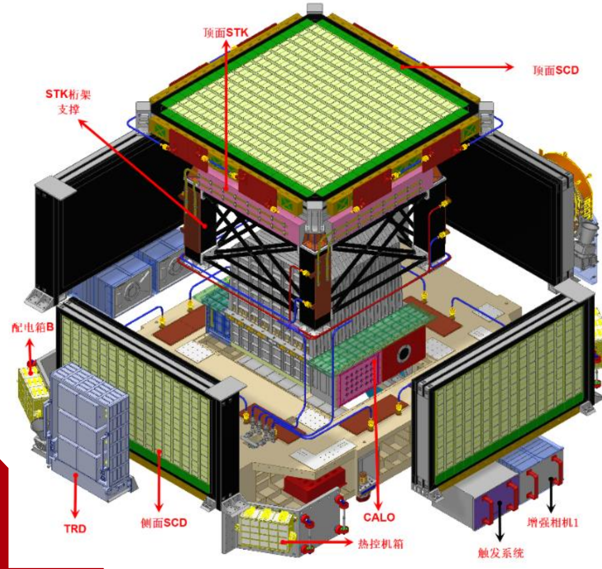
2024/10/19

High Energy cosmic-Radiation Detection facility...

SHANDONG UNIVERSITY

- HERD is a space particle astrophysics experiments, will run in the Chinese Space Station for more than ten years.
- The science goals are precision measurement of cosmic ray electron flux and dark matter search, origin of cosmic rays and high energy gramma rays all-sky survey and monitoring.
- HERD Detector consists of five sub-detectors: three-dimensional **Calorimeter**, **FIber Tracks**, **Plastic Scintillator Detector**, **Silicon Charge Detector** and **Transition Radiation Detector**.
- The core scientific capabilities of HERD will maintain a significant international lead ship for a long time.





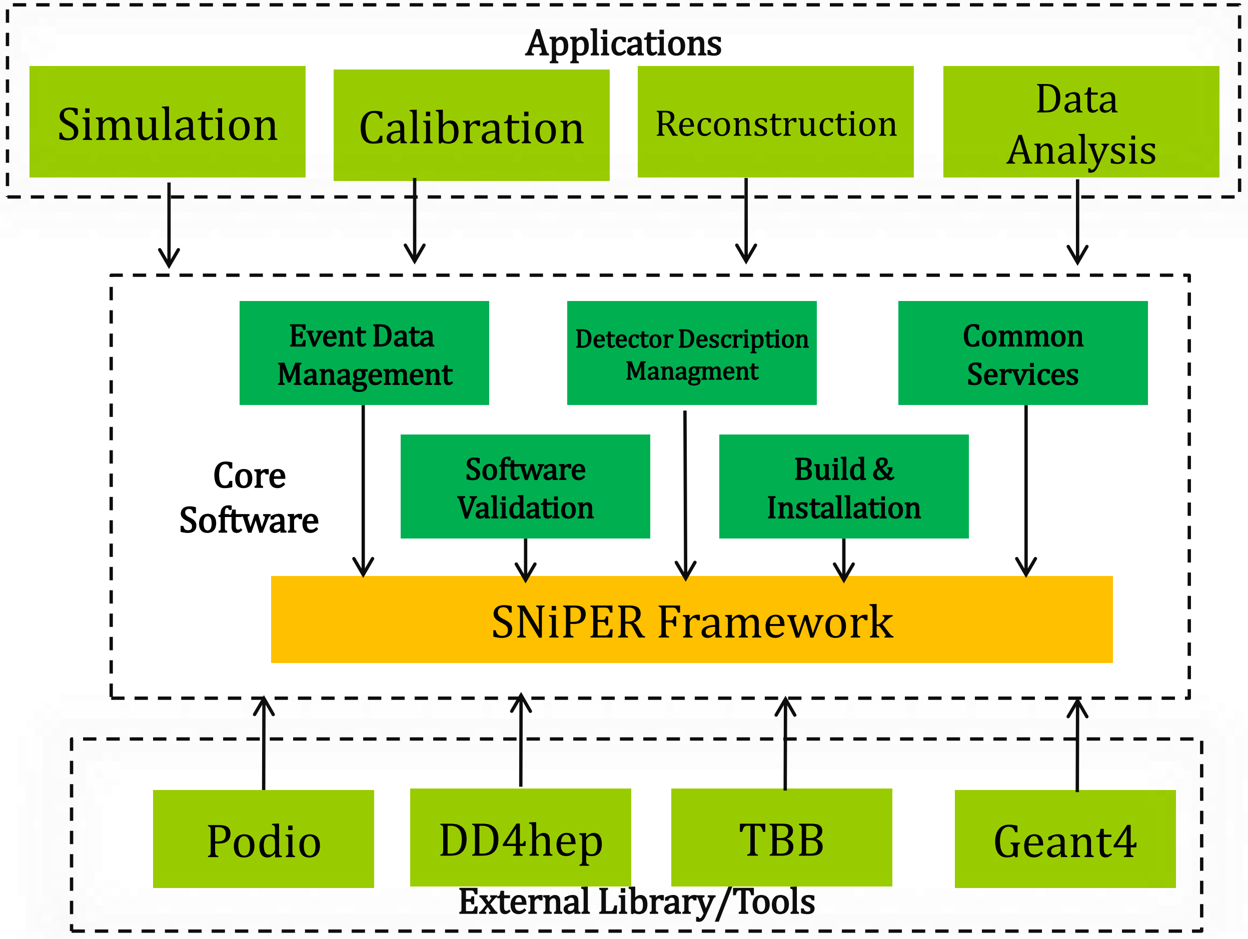
5 side FIT

1 side STK

Overview of HERD Offline Software System

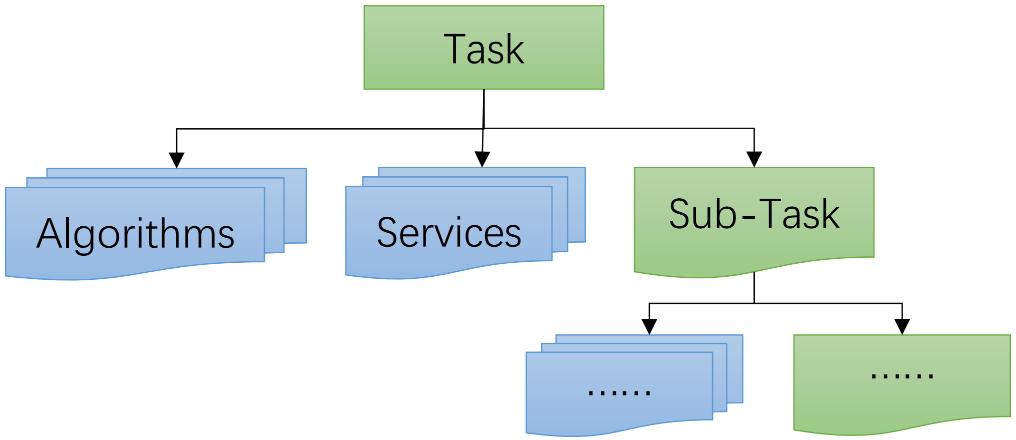
- HERD Offline Software (HERDOS) is designed for detector design, MC data production and physics analysis, and provides a common platform for user to develop and perform analysis tasks.

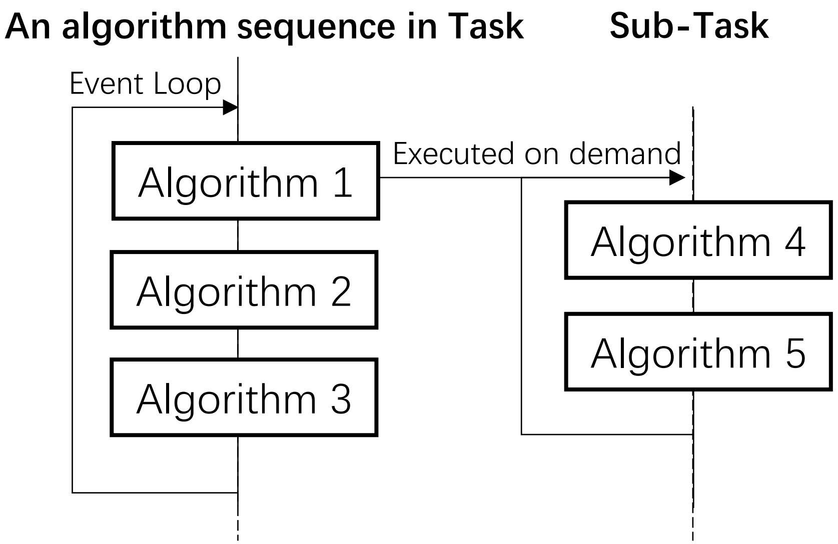
- The application software is developed based on the core software, and relies heavily on the functionalities provided by the core software.
- The core software provides the common functions and key technologies:
  - Detector data and event data management
  - Detector and event display
  - Support of parallel computing and machine learning
  - Common services
- The core software is partially based on the Key4hep and the modern software stack.



Underlying Framework： SNiPER

- HERD is a **lightweighted**, **long lifecycle** space station experiment
  - The underlying framework should be light as a sparrow, yet complete in every part, and be of great performance
- Software for non-collider experiments (SNiPER)
  - Adopted by JUNO, LHAASO, nEXO, STCF
  - Provide basic functionalities of event loop, application interface, job configuration, logging etc.
- Advantages of SNiPER
  - Lightweighted, efficient, highly extendable
  - High cohesion & low coupling design
  - Flexible event loop control
  - Flexible processing chain can be built on demand
  - Multi-task mechanism, powerful parallel support
  - C++/Python hybrid programing



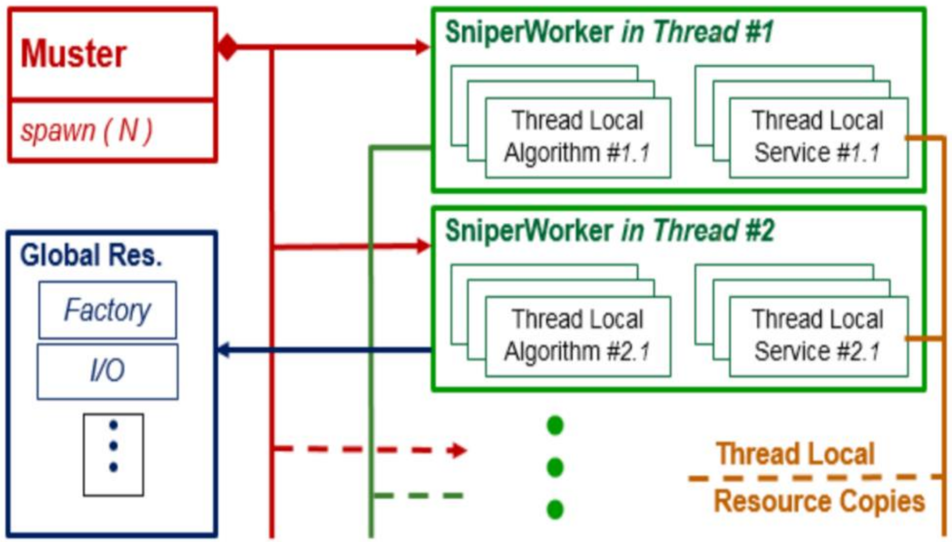


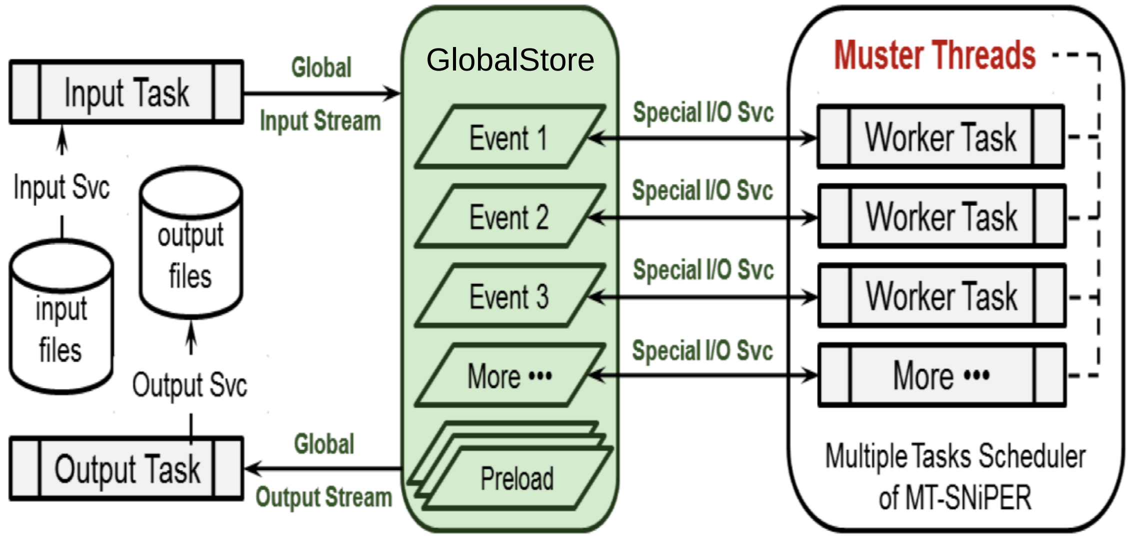
Requirement for Multithreading ...

- **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 helion costs **~20h**
  - Memory consumption is huge, often causing job gets killed
- Applying concurrent simulation can:
  - **Reduce absolute time cost** of simulating heavy particles
  - **Decrease the total memory consuming** by sharing objects in memory
    - Geometry (TGeoManager costs ~ 1GB momery), common services, I/O Buffer, physics list ...
- Multi-level of multithreading can be applied
  - **Event level** (between events): multiple events are processed concurrently
  - **Track level** (inside an event): one event is processed with multiple threads
    - For example, different tracks are simulated concurrently

MT-SNiPER …

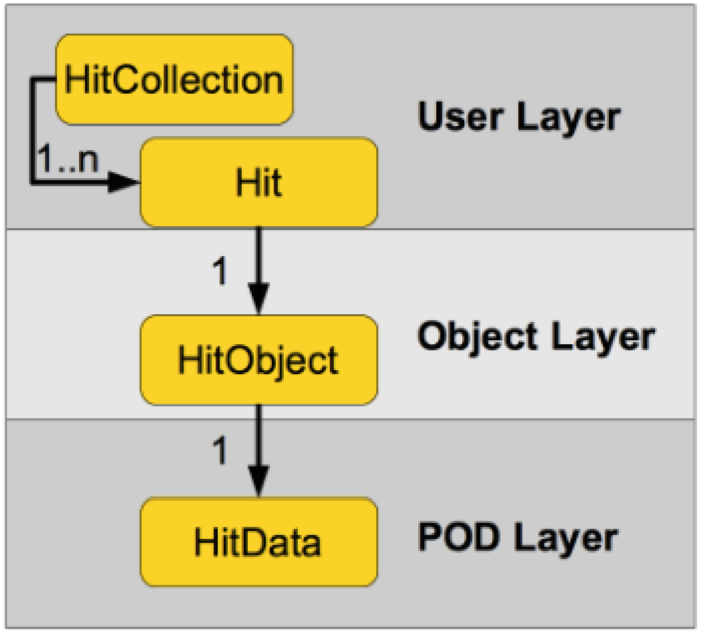
- SNiPER provides simple interfaces for building the event-level multithreaded applications
  - Based on Intel TBB
  - SNiPER Muster (Multiple SNiPER Task Scheduler) works as a thread pool/scheduler
  - A GlobalStore is developed to support parallel event data management
  - Data I/O is bound to dedicated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





Event Data Model Based on Podio…

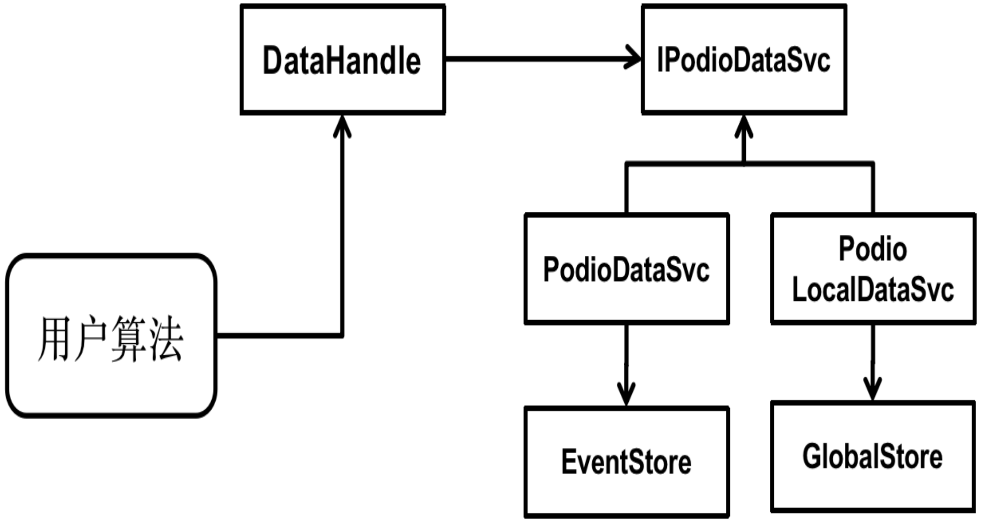
- Event data model (EDM) is the crutial part of the framework
  - Define the structure of event data in memory and in data files
  - Construct relationship between EDM objects(tracks-hits)
- HERDOS chosed podio as the toolkit for EDM definition
- Podio: common EDM toolkit (AIDA project, used by FCC, CEPC, ILC, STCF)
  - Generate C++ code automatically from YAML files
  - Support analysis in ROOT and Python
  - Support concurrent access to event data
  - Well handled relationship between EDM objects



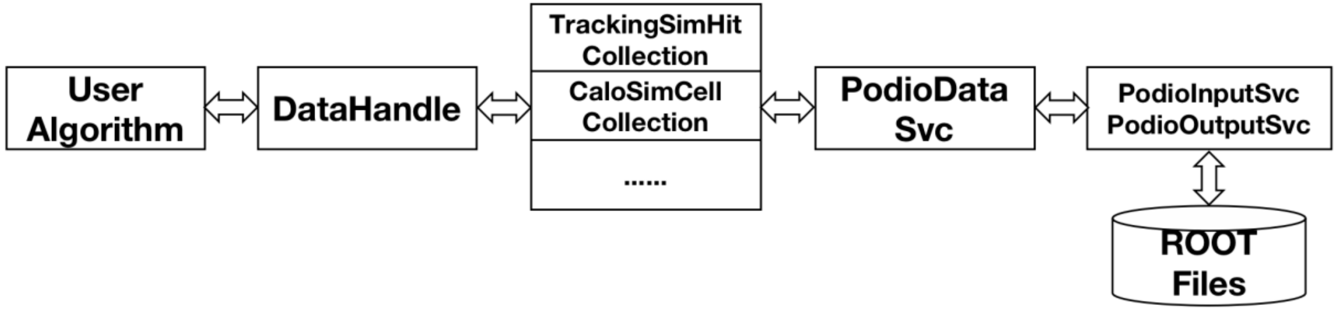
Official EDM classes can be extended on reasonable demand, users could define their own EDM classes for analysis

Event Data Management . . .

- Event Data Management (DM) system manages event data in memory, provides interfaces for user applications and handles data I/O.
- HERDOS extend SNiPER DM system based on Podio
  - PodioDataSvc: manage podio:EventStore (serial)
  - PodioLocalDataSvc: access GlobalStore (parallel)
  - PodioInputSvc: data input
  - PodioOutputSvc: data output
  - DataHandle: interface for user to access data

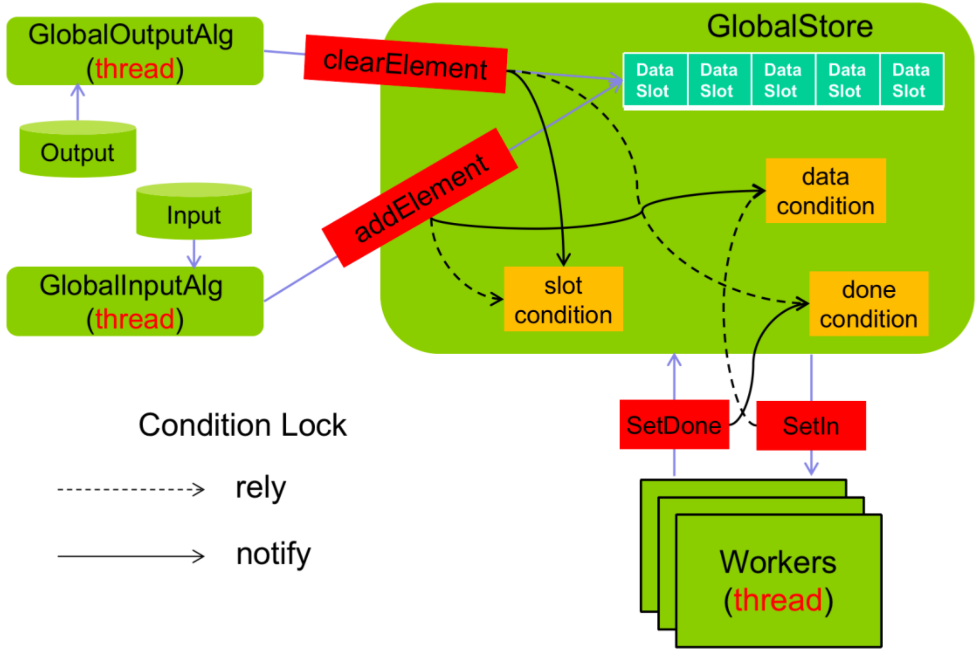


- Event data and user application are completely decoupled



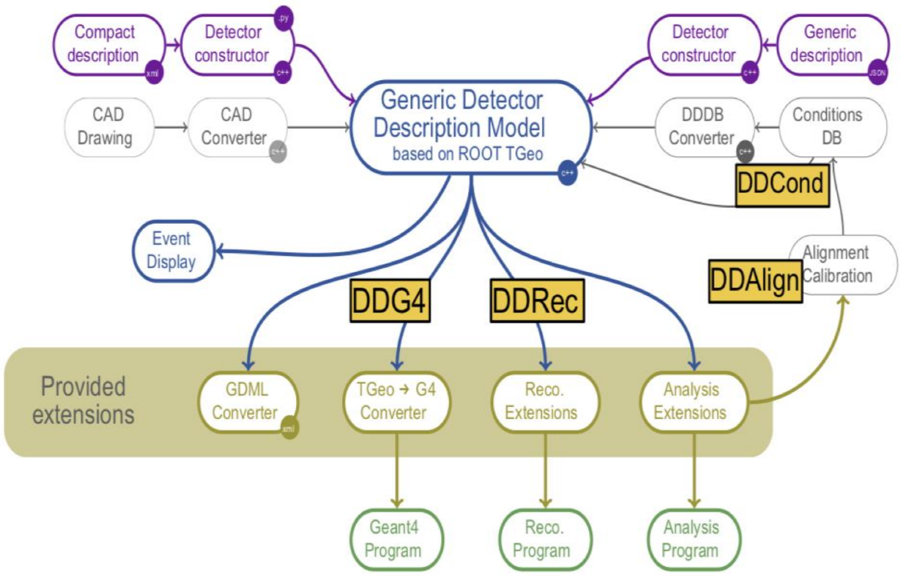
Parallelized Event Data Management…

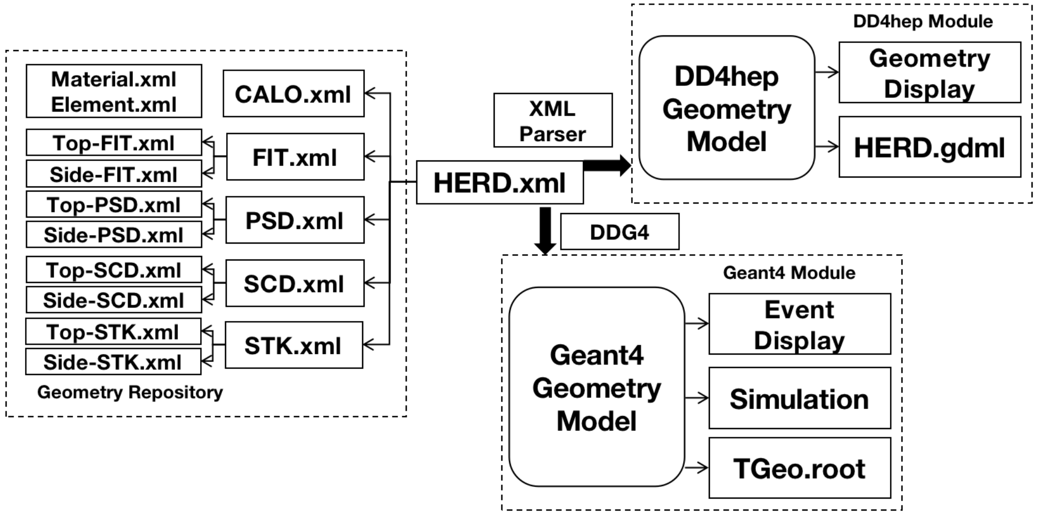
- To enable parallelized data processing, a GlobalStore is developed and implemented based on podio
  - Re-implement podio::EventStore to cache multiple events (each within one data slot)
  - I/O services are binded to dedicated I/O threads, to ensure performance and flexible post- or pre-processing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



Geometry Description Management…

- A powerful detector description management system is necessary across the full offline data processing workflow
- DD4hep is chosen as the core of HERDOS detector description management system
  - Define geometry in xml files , using TGeo objects as a unified memory format, and provide multiple plugins.
- Full HERD and beam test geometries are defined in XML files
  - Elements, materials defined in shared files then composed together
  - Sub detector defined separately, with independent versioning scheme
  - The version scheme allows switching detector description during run time
  - Complex geometry (including the space station) from CAD format will be implemented





Detector Geometry Service ...

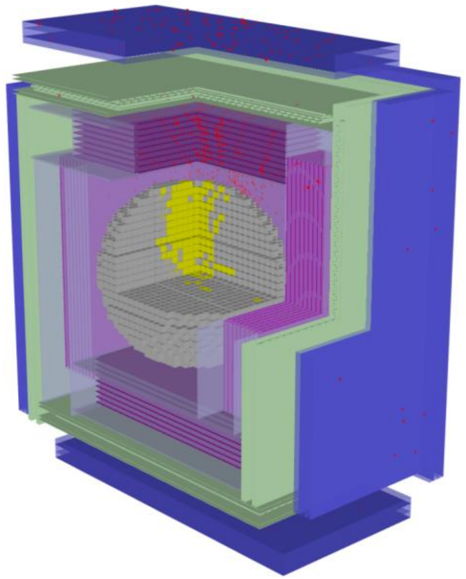
- To provide an easy-to-use interface for applications, **GeometrySvc** is implemented to integrate and provide various detector description information:
  - Conversion between geometry description formats (XML, CAD, Geant4, ROOT, GDML, ...)
  - Global-Local coordinates conversion (cellID)
  - Coding scheme conversion (cellID, volumeID, cellcode)
  - Calculate position, dimension of all detector volumes
  - Calculate track length in physics volumes
  - Provide interface to get physical volume, placed volume, logical volume
- These functionalities are actively used in simulation, digitization and reconstruction applications

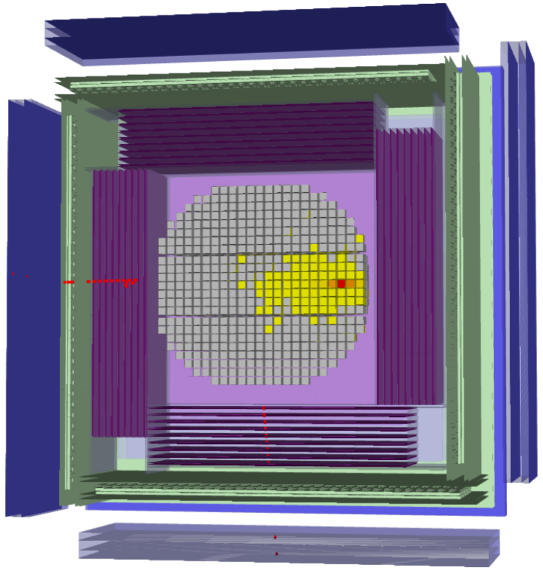
```
// Get geant4 geometry information
dd4hep::sim::Geant4GeometryInfo* getGeoInfo();
// Get geant4 physical Volume
G4VPhysicalVolume* getPhyVol();
// Get geant4 magnetic field
G4MagneticField* getMagField();
// Get dd4hep detector instance
dd4hep::Detector* getDetDesc();

// 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 cellcode);

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

// 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);
```

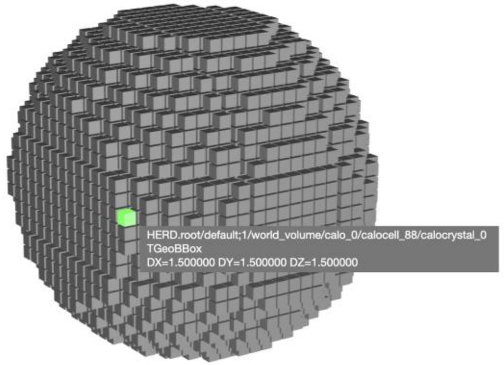
Detector and Event Display

- Visualization of the detector and event data is important for designing and optimizing the detector, debugging the offline software, carrying out physics analysis, monitoring and outreach etc.

- HERD Event visualization (HERDEvE) is being developed
  - User interface and 3D display based on WebGL
  - 3D engine and graphic library based on Three.JS
  - Geometry information from detector description based DD4hep (XML), and event data read from podio
  - Reducing 3D motion lag by the multi-threading capabilities of Web Worker framework
  - Using the Vue.js HTML5 development framework to implement the Web interface

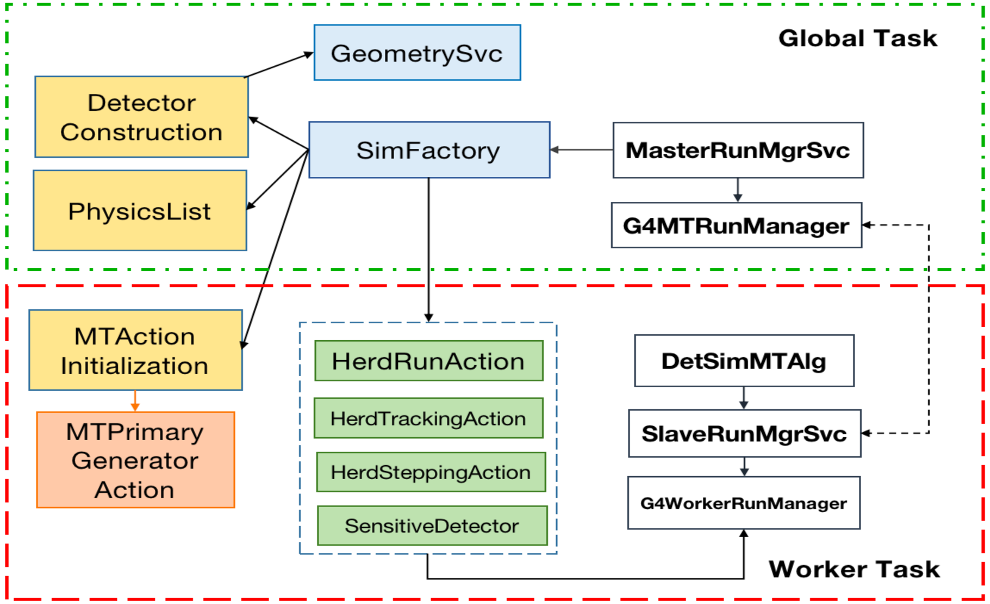


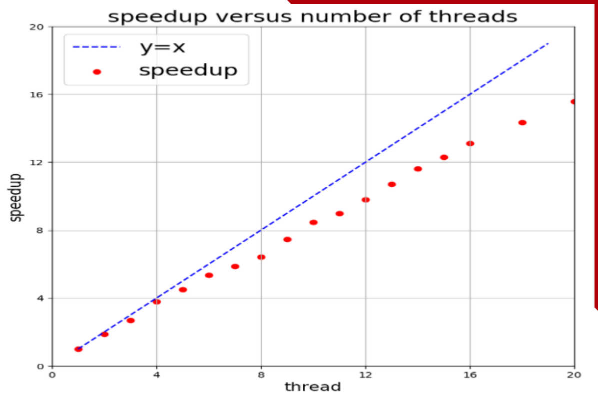


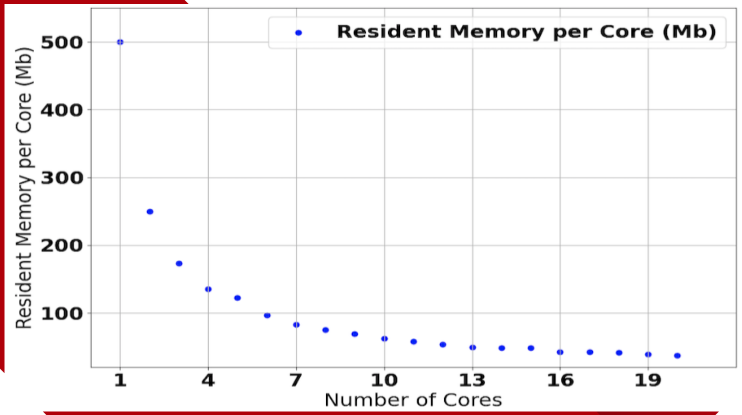


Parallelized Detector Simulation…

- HERDOS integrates SNiPER, DD4hep, Geant4 and podio to provide a unified detector simulation interface.
- Based on SNiPER, interface implements modular design to ensure that various development works do not interfere with each other.
- Based on the MT-SNiPER and parallelized DM system, the event-level parallelized detector simulation is developed. Maximize the reuse the modules implemented in serial mode.
  - Simulate events concurrently in multiple threads
  - Basic performance tests show promising scalability

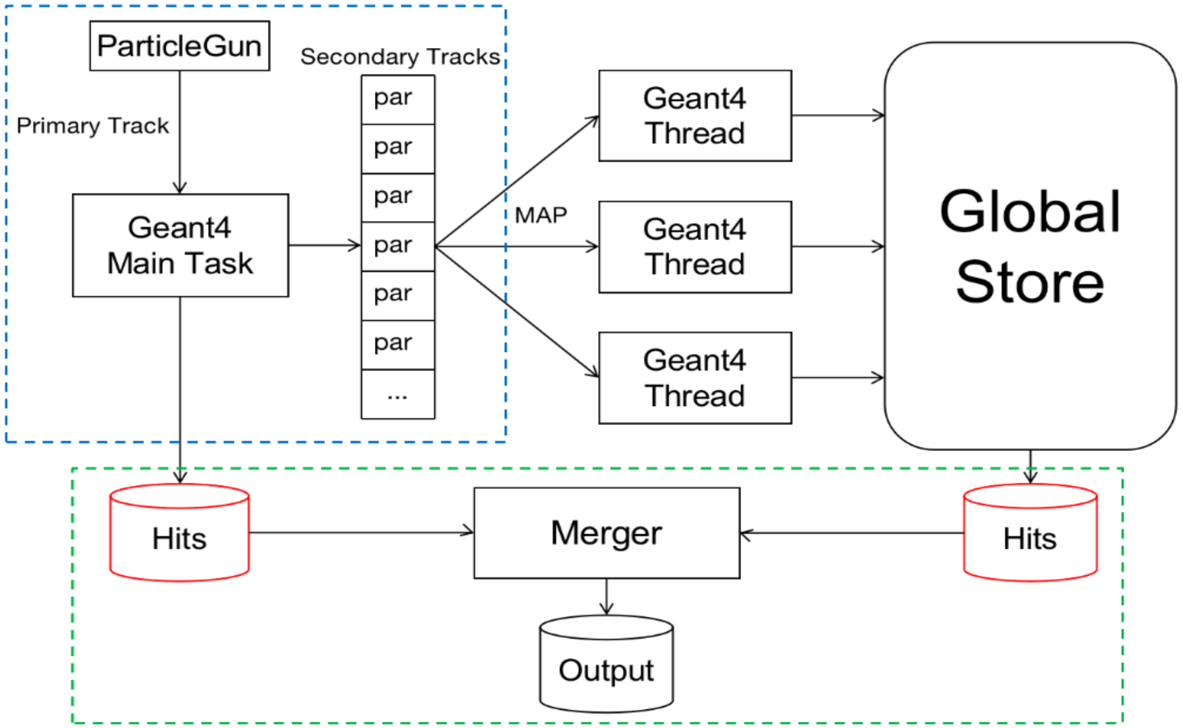


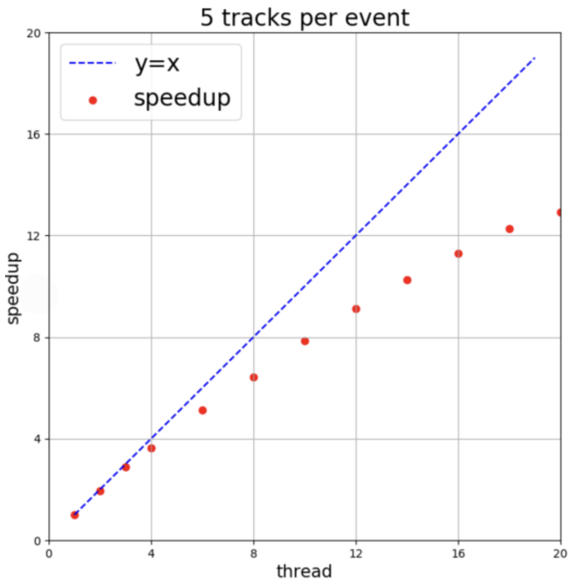


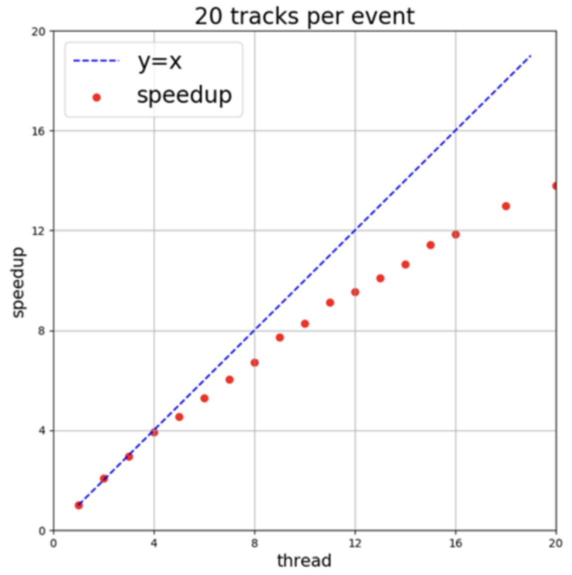


Track-level…

- 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

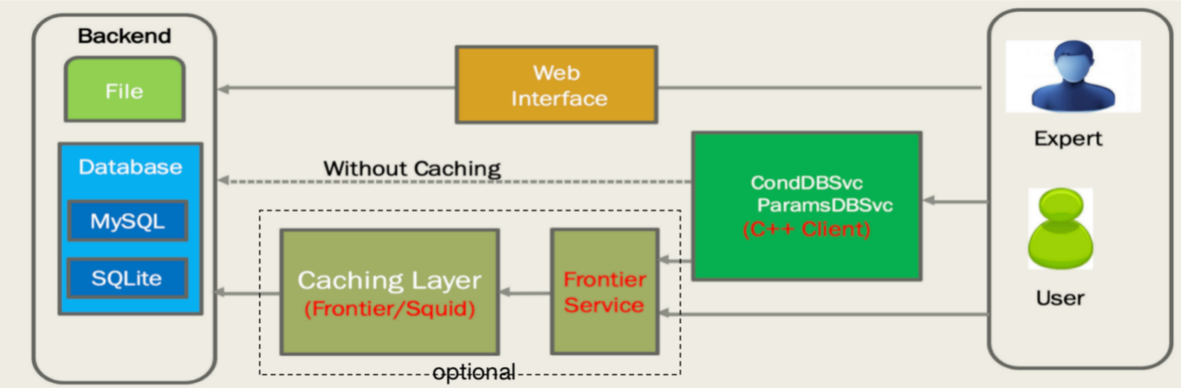


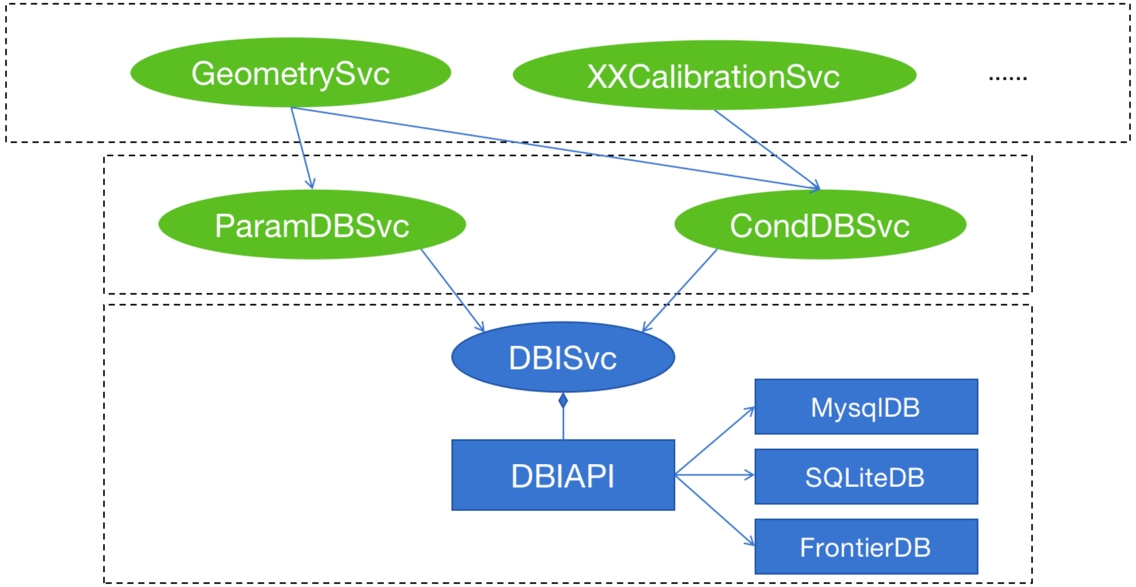




DataBase System ...

- Overview of database system
  - Web interface, C++ client interface, Frontier service





- Modular architecture of database service
  - DBIAPI provides a unified interface to different backends, including Frontier, MySQL, SQLite
  - DBISvc acts as a manager to provide the underlying database connection
  - ParaDBSvc and CondDBSvc are used by the applications, provides static parameters (load once at initialization) and the conditions data (update automatically)
  - GeometrySvc, CalibrationSvc, AlignmentSvc provide specific parameters for services
- Implement long-term orderly management of calibration parameters using GlobalTag, Tag and IOV

Summary...

- Introduce the basic design and functionalities of HERD Offline Software system
  - Developed partially based on Key4hep
  - Many components are extended specifically for HERD, but also re-usable by other experiments
- Introduce the design and implement of HERDOS
  - Underling framework: SNiPER
  - The design of EDM based on podio and implementation of DM system through the integration of podio and SNiPER
  - The implementation of parallelized DM system based on SNiPER and TBB, the development of GlobalStore
  - The geometry management system based on DD4hep that provides consistent detector description, an easy-to-use interface for applications
  - The parallelized detector simulation framework， including both event-level and track-level parallelism.
  - The design of database system to manage calibration parameters
- HERDOS is operating effectively to support the design of the detector, as well as the exploration of its physics potential.

Thank You For Your Listening !