Introduction to Lodestar

(LHAASO Offline Data Processing Software Framework)

Xingtao HUANG, Xueyao ZHANG, Teng LI

Shandong University

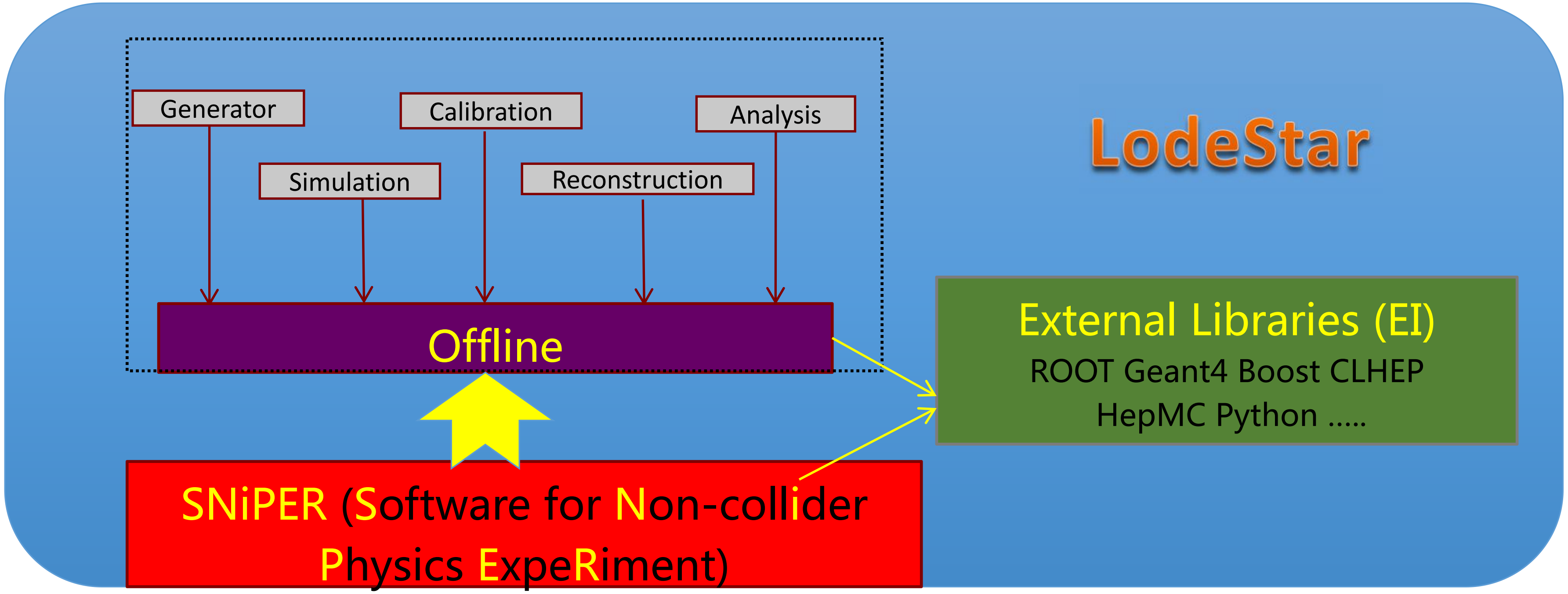
2016.8.15

Outline

- Overview of Lodestar
- Introduction to SNiPER
- Development Status
- Summary

Overview of LodeStar

- **LodeStar:**
  - LHAASO Offline Data Processing Software Framework
- **Constituents of LodeStar:**
  - offline: specific to LHAASO Experiments
  - SNiPER: underlying framework
  - External Libraries: frequently used third-party software or tools



Functionalities of Lodestar

- **Management of the offline data processing procedures**
    - Generator, simulation, reconstruction, analysis etc.
    - Modular management
    - Process/Sequence management
- **Management of the event data**
    - Event Data Model
    - Data Input/output
    - Data storage
- **Providing common services or tools during processing**
    - HisogramSvc, RandomSvc, DatabaseSvc, etc.
- **Providing friendly user interfaces**
    - Easy to develop user code
    - Configure jobs

Software Environments

- **Programming Language： hybrid programming of C++ and Python**
  - C++: main part implementation
  - Python: job configuration interface
- **Packages management tool: CMT (Configuration Management Tool)**
  - Help developers to manage packages easily
  - Help users to setup the environment for running the application easily
- **Operation System: Linux**
  - Official support: Scientific Linux (Now SL 6+ gcc4.4)
  - More OS will be tested and supported according to needs
- **Codes Management: SVN**
  - Keep history of codes evolution
  - Synchronization and sharing between developers
  - Tag and release

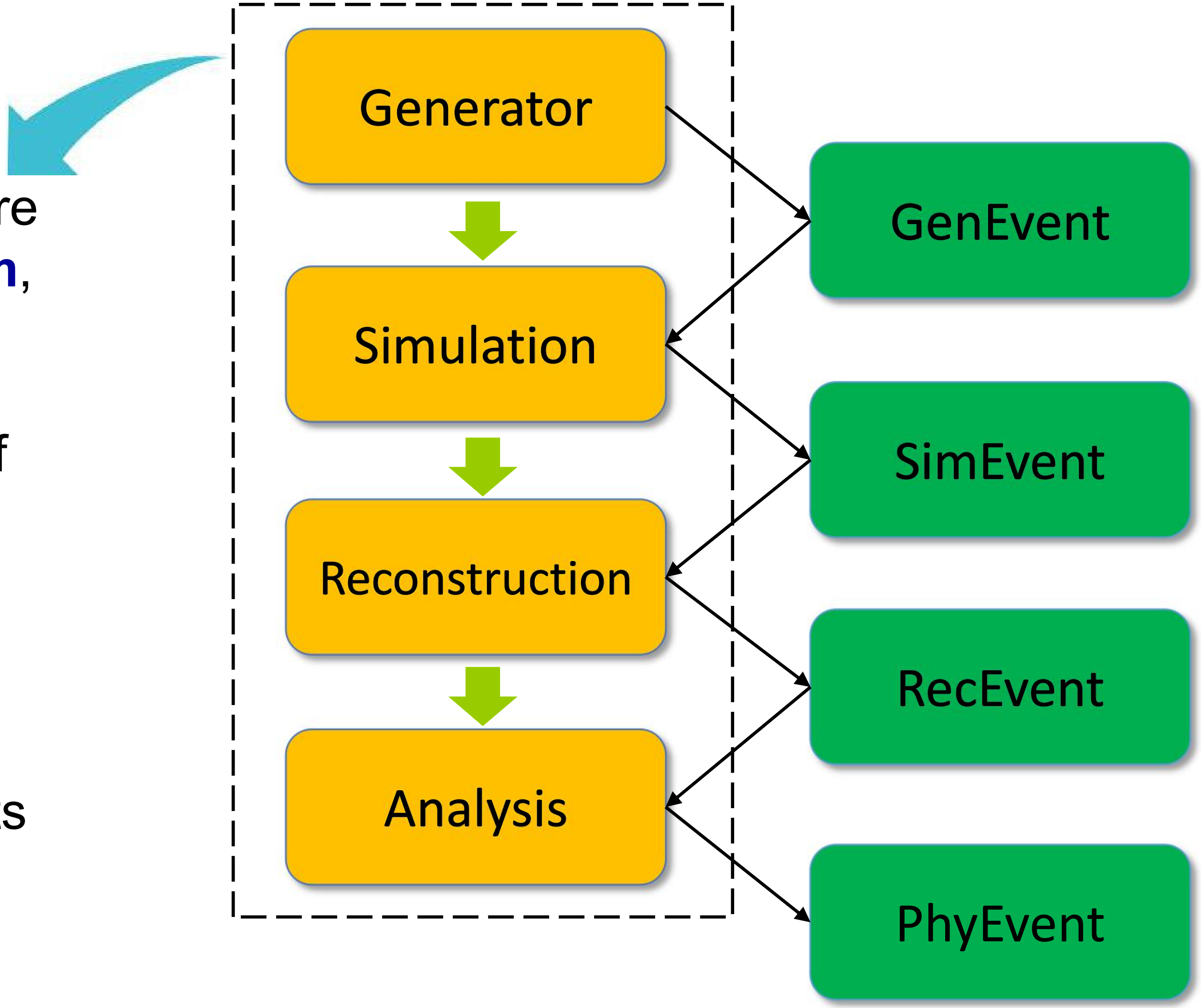
Procedure of Offline Data Processing

- Management of the offline data processing, including generator, simulation, reconstruction, analysis etc.

- The data processing procedures are developed based on the **Algorithm**, **Service** and **Tool** in SNiPER

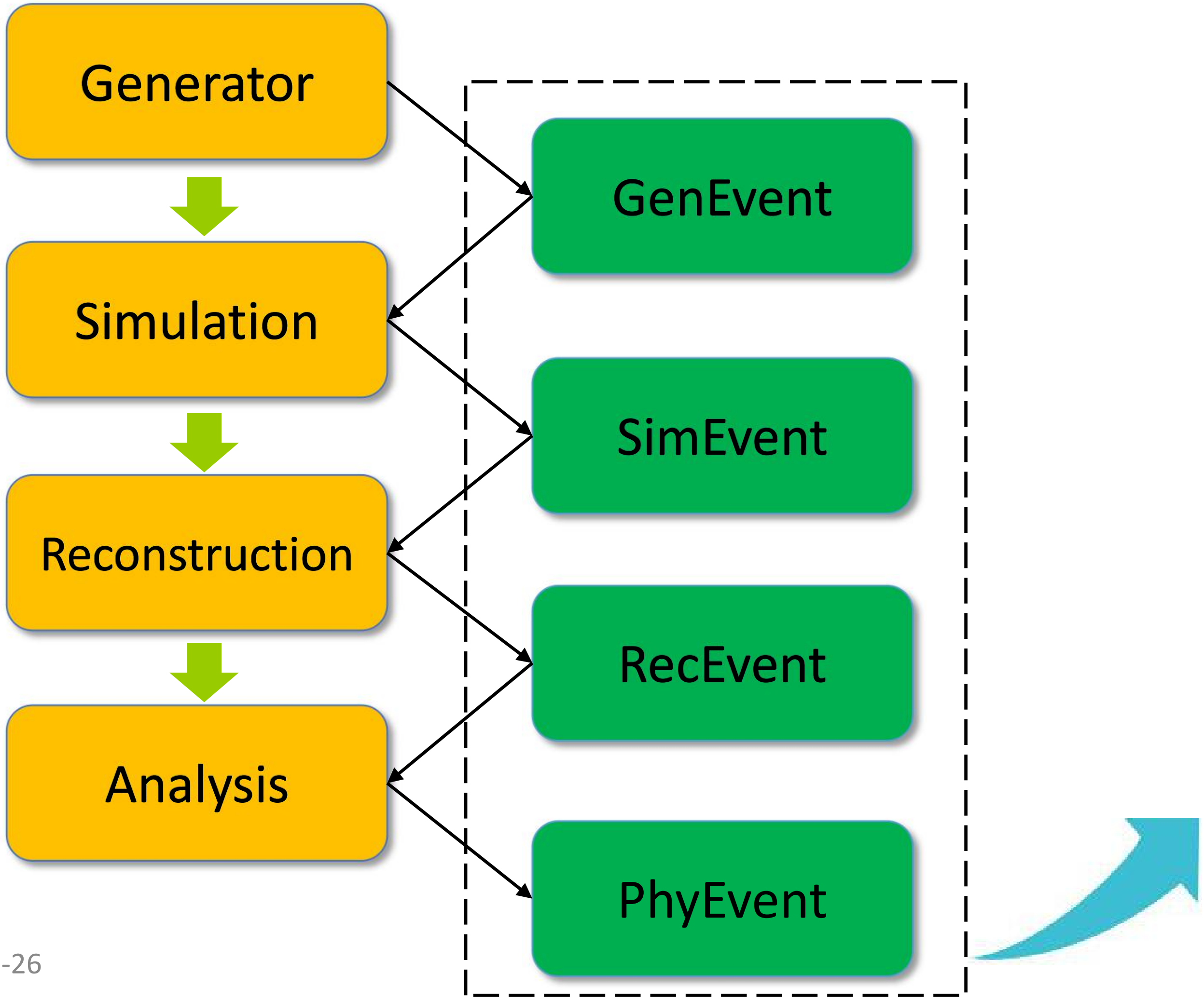
- Each procedure usually consists of one or several algorithms, e.g.
  - direction reconstruction
  - energy reconstruction

- Each procedure reads event data from previous step and produces its event data for the next one



Procedure of Offline Data Processing

- Management of the offline data processing, including generator, simulation, reconstruction, analysis etc.



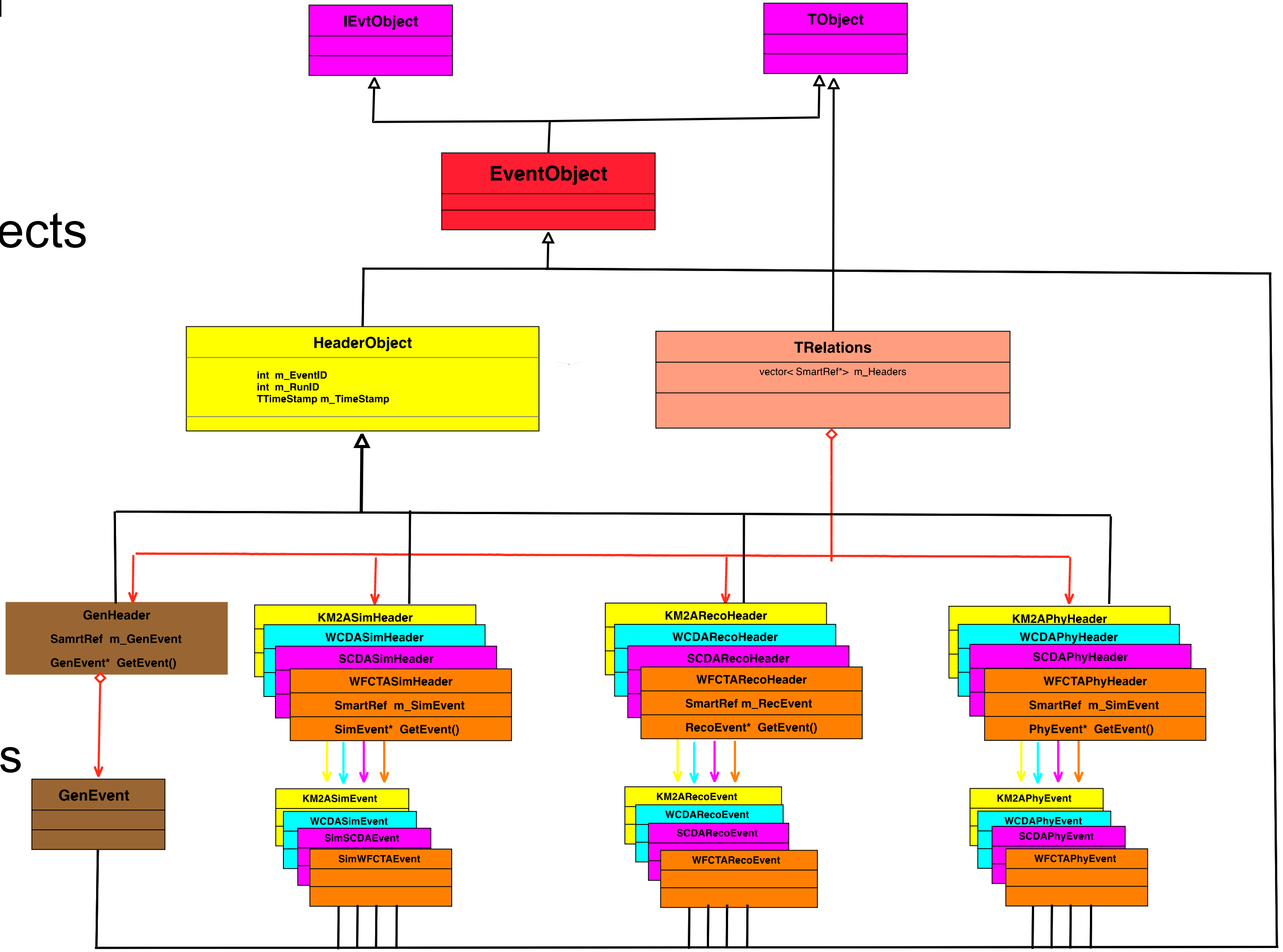
- The event data produced by algorithms is saved in **Event Data Model** objects
- Event Data Model objects are stored in a place named as **DataStore**
- Data Model objects can be converted into persistent form and saved into files

Design of the LHAASO Event Data Model

- LHAASO EDM is designed based on ROOT
- EventObject inherits from TObject
- Each process defines their EDM Objects
- EDM for each process is split into two parts in order to achieve quick event selection:
  - HeaderObject

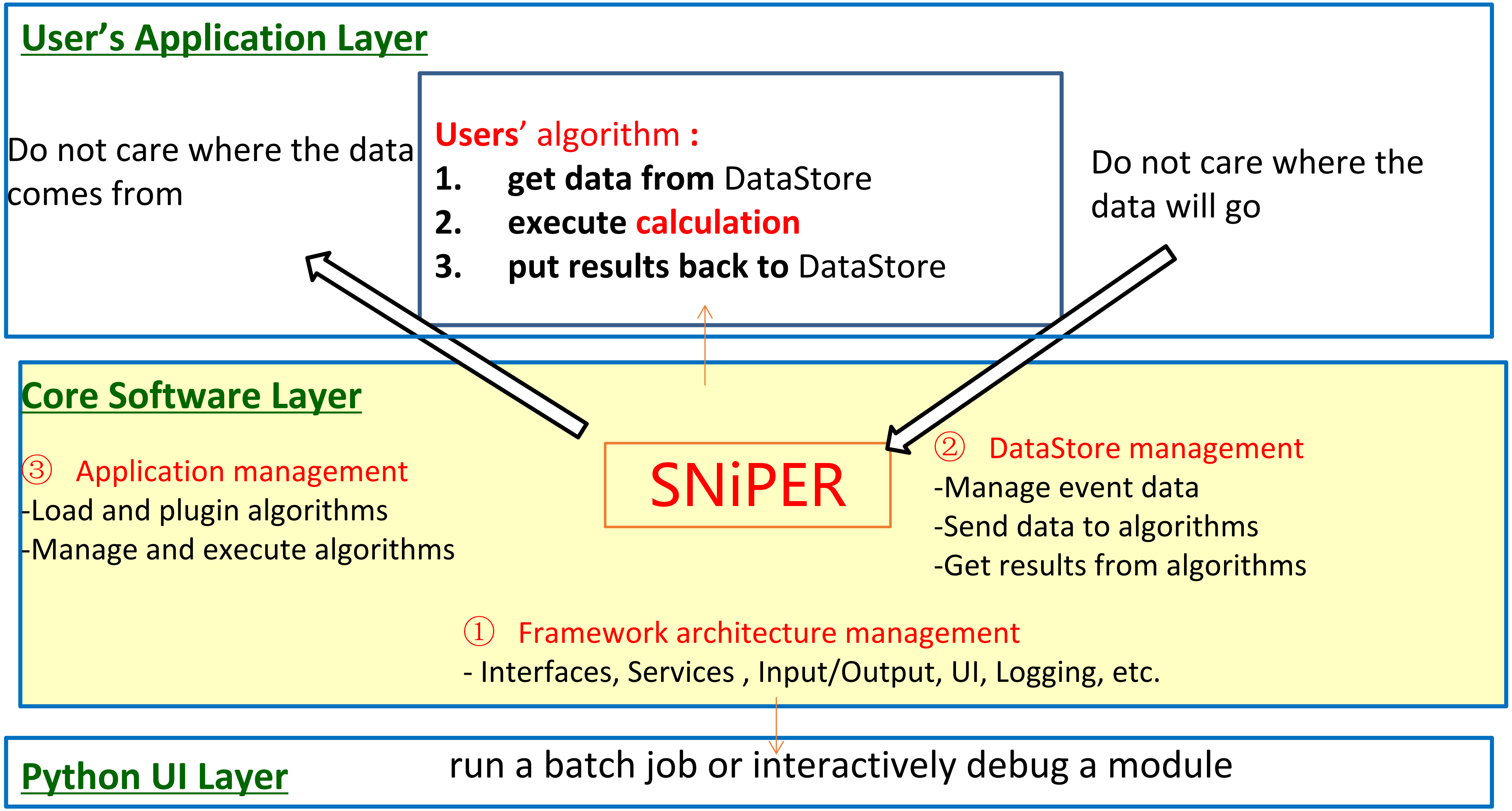
  For the tag information
  - EventObject

  For the full event data
- TRelations: Matching between Headers (optional)



Introduction to SNiPER

- What does SNiPER do?



User’s Application Layer

Do not care where the data comes from

Users’ algorithm :

1. **get data from** DataStore
2. **execute calculation**
3. **put results back to** DataStore

Do not care where the data will go

Core Software Layer

③ Application management
-Load and plugin algorithms
-Manage and execute algorithms

SNiPER

② DataStore management
-Manage event data
-Send data to algorithms
-Get results from algorithms

① Framework architecture management
- Interfaces, Services , Input/Output, UI, Logging, etc.

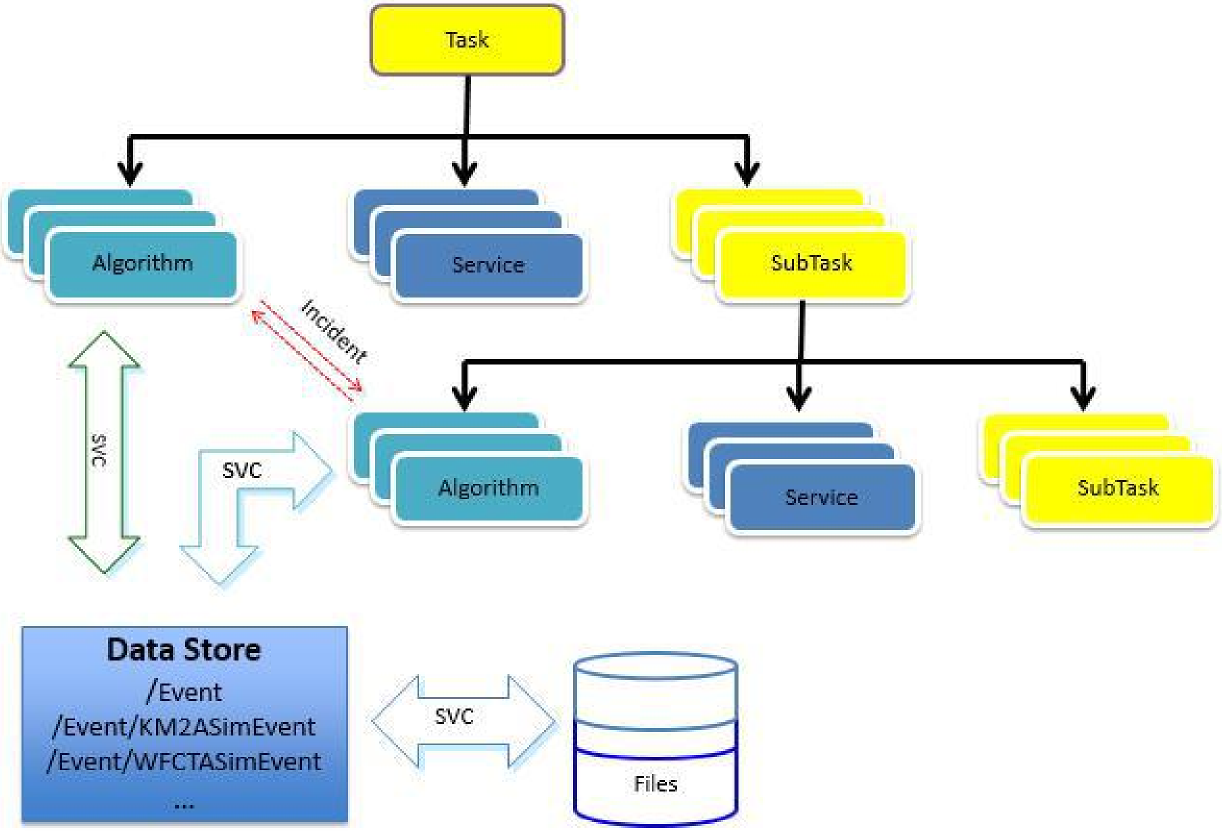
Python UI Layer

run a batch job or interactively debug a module

Main components of SNiPER

From Users' point of view:

- **Algorithm**
- **Service**
- **Task**
- **Incident**
- **DataStore**



◆ Algorithm, Service and Task follow modular design

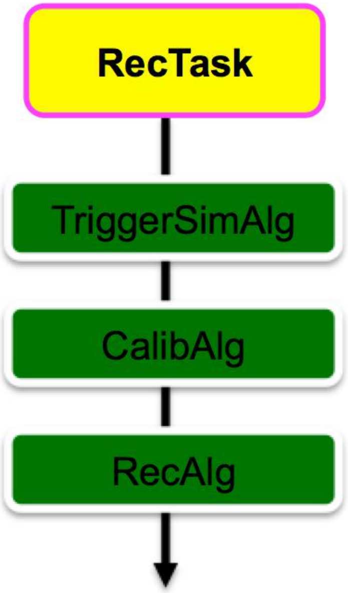
⇨ Dynamically Loadable Element (DLElement)

⇨ Low couplings between each other

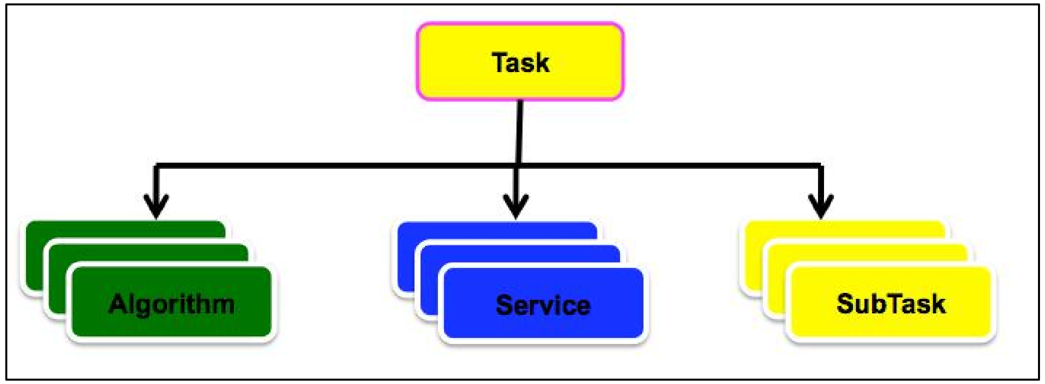
⇨ Support parallel development of applications

Algorithm and Service

- **Algorithm**: The smallest unit of users' codes:
  - **perform event calculation , i.e.**
    - **Position reconstruction**
    - **Correlation analysis**
  - **SNiPER provides user interface (AlgBase)**
    - **User's algorithm must inherit from AlgBase**
  - **One data processing (or Task) consists of one or more algorithms**

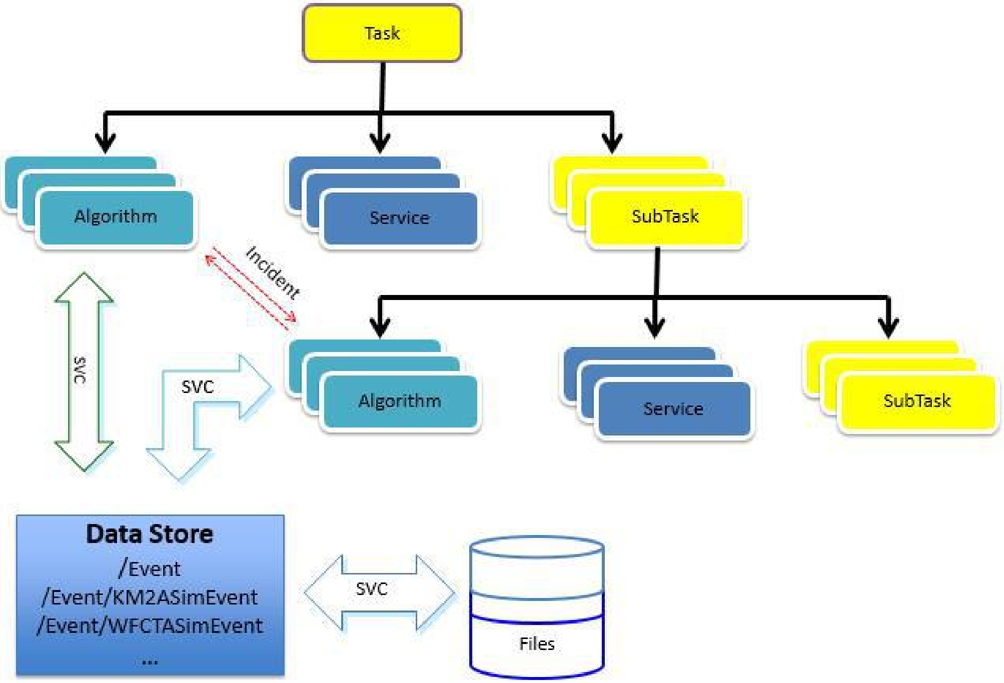


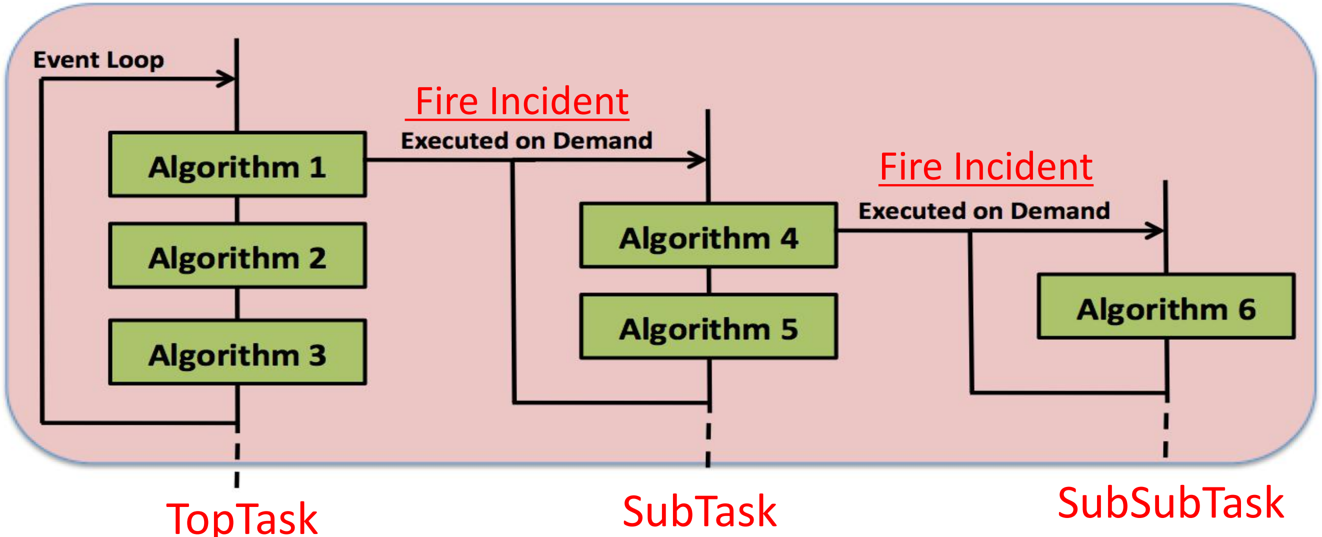
- **Service:** Usually a piece of codes for common use:
  - **Histogram Service**
  - **Random Service**
  - **Geometry Service**
  - **ROOT Input/Output services, etc.**
  - **User interface, SvcBase , is provided by SNiPER**
    - **New services must inherit from it**



Task

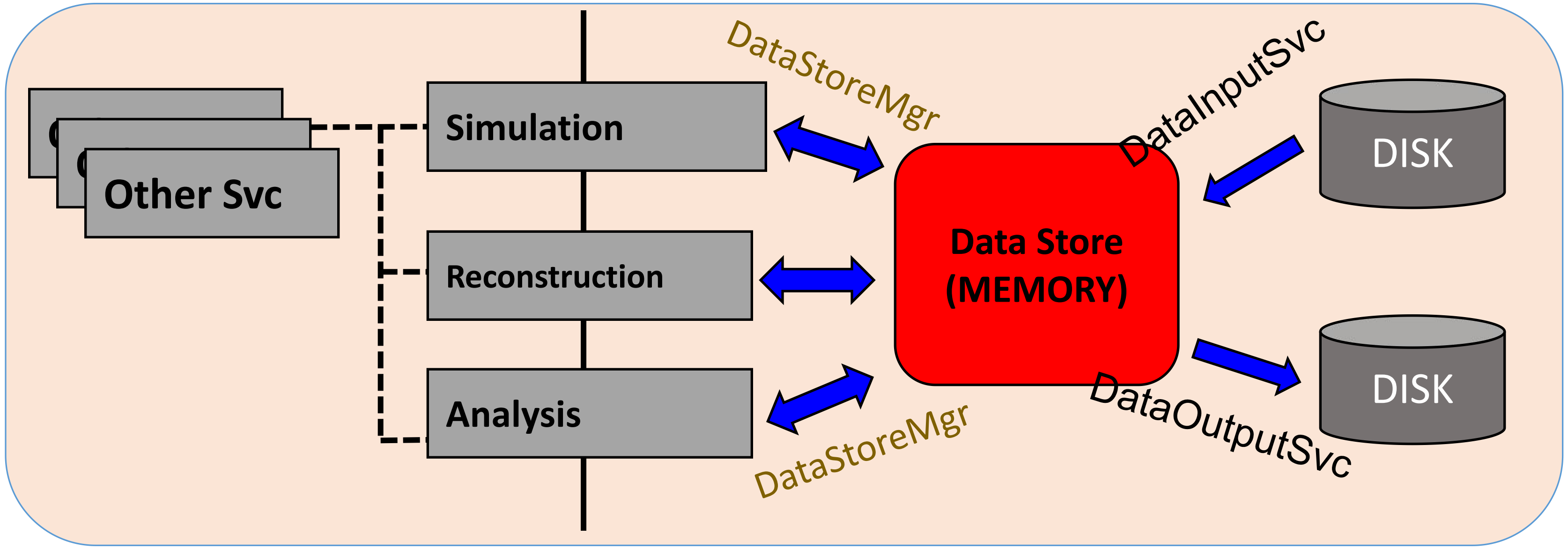
- **A lightweight application manager:**
  - **Manage its Algorithms/Services**
  - **Perform sequential Algorithm execution**
  - **Manage DataStore and input/output systems etc.**
- **A job may have more than one Tasks**
  - **TopTask and SubTask**
- **Each task can be configured individually**
- **Both sequential and jump executions are implemented**
  - **SubTask can be executed by firing Incident**





Data Store and Data Input/output

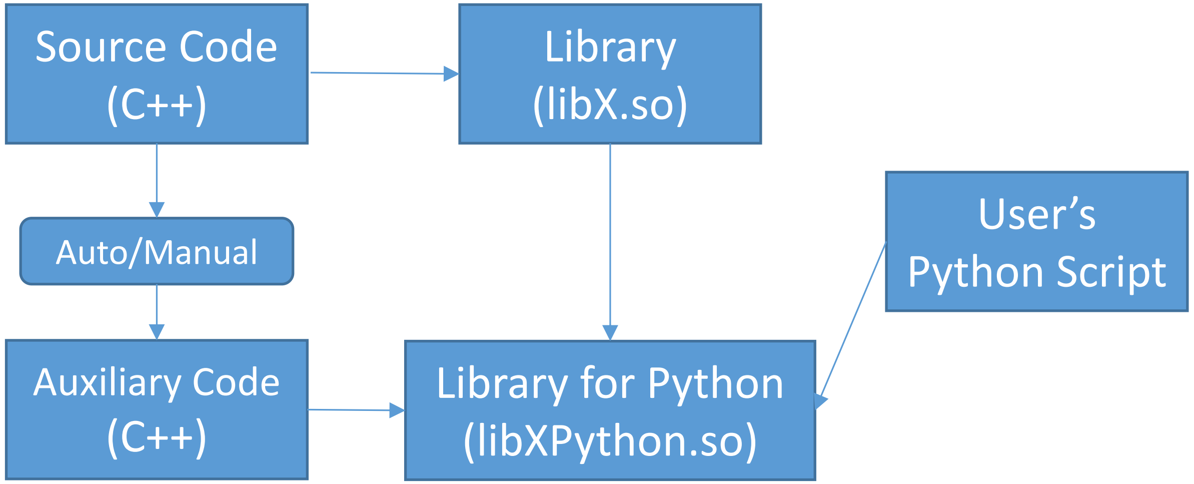
- DataStore is the dynamically allocated memory place to hold event data
  - Managed by SNiPER service: DataStoreMgr
- Applications (in terms of algorithms) get/put event data from DataStore
  - Smart pointers are provided to perform
- DataStore is automatically configured with Data I/O Service
  - Before/after event processing, event data will be input/output

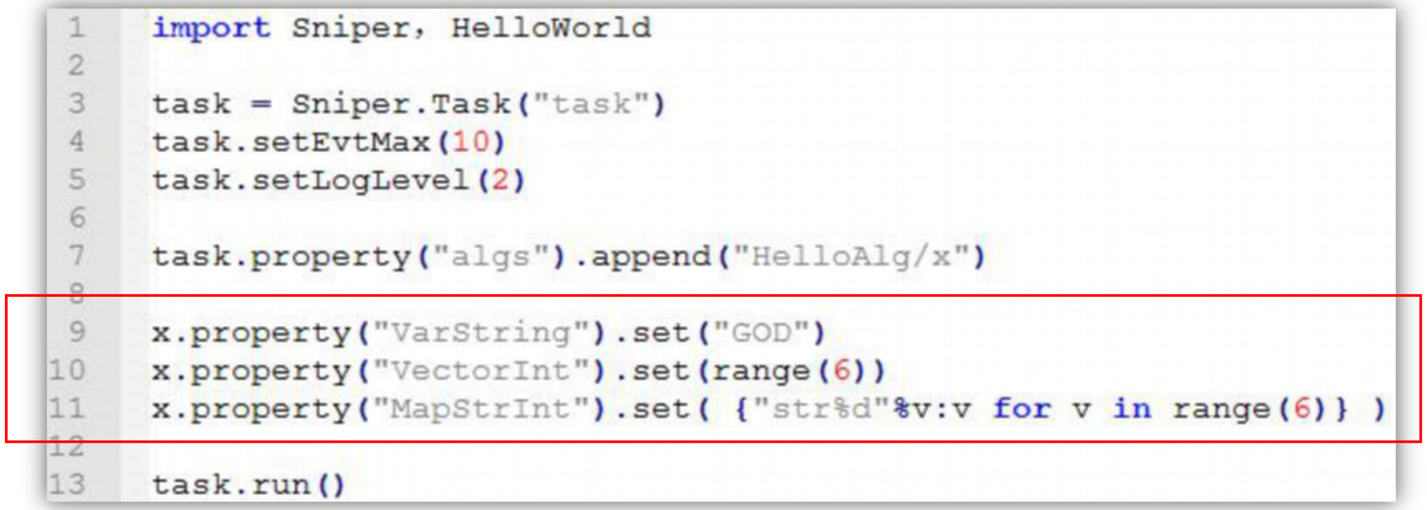


Python User Interface

- Python binding is used to configure/run SNiPER jobs
  - Based on Boost.Python
  - Take advantage of the flexibility of Python
  - In place of the txt job option files
- Task, Algorithm, Service are all configurable in Python
  - Import SniperPython modules
  - Create Task
  - Set up Algorithms, Services, etc.

    Use property to configure the run time variables
  - Invoke the task





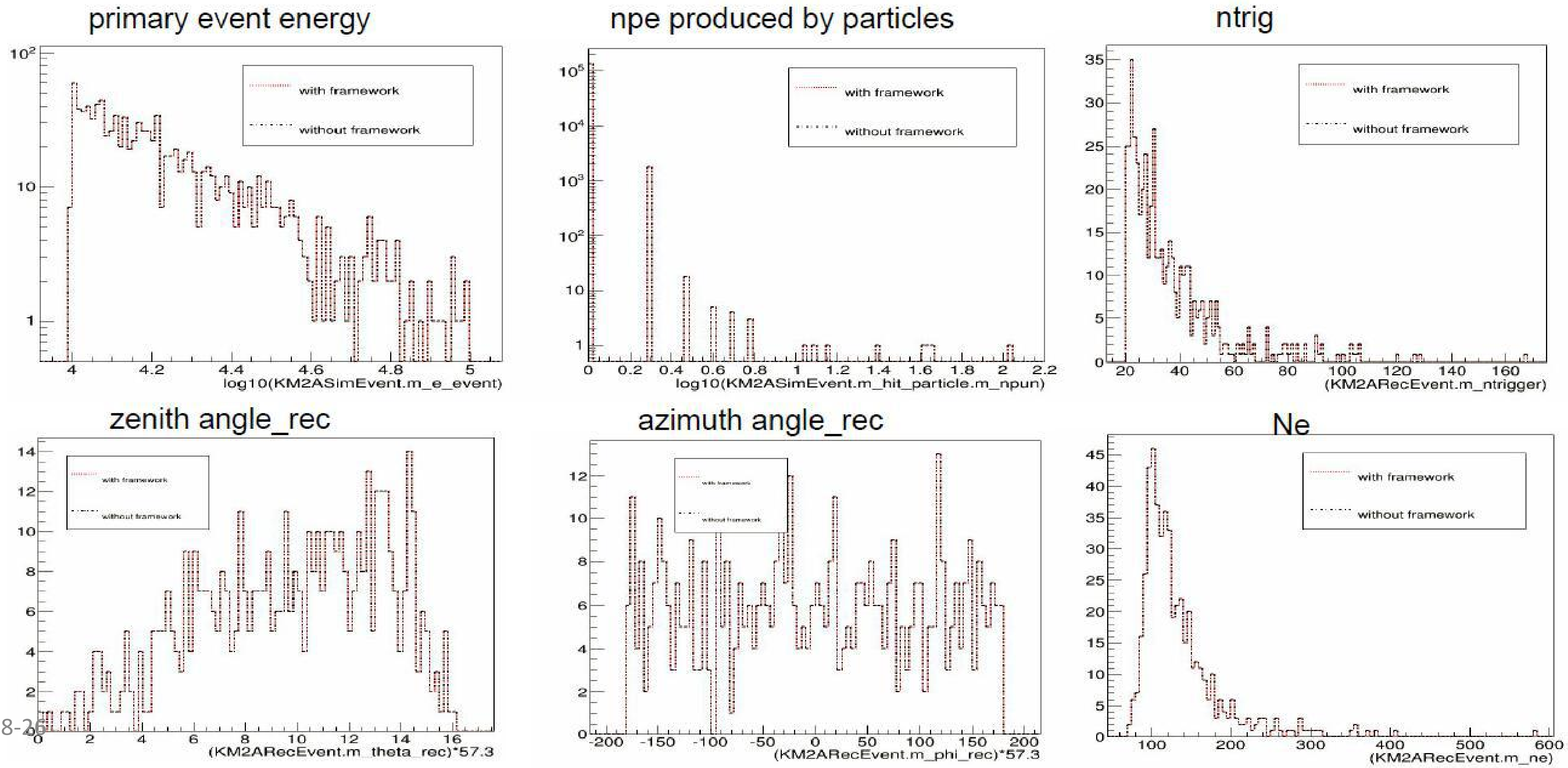
```
import Sniper, HelloWorld

task = Sniper.Task("task")
task.setEvtMax(10)
task.setLogLevel(2)

task.property("algs").append("HelloAlg/x")

x.property("VarString").set("GOD")
x.property("VectorInt").set(range(6))
x.property("MapStrInt").set( {"str%d"%v:v for v in range(6)} )

task.run()
```

Development Status of Lodestar

- Lodestar has been built on SNiPER
- Basic tools for the offline data processing applications are ready to use:
    - Event Data Model
    - DataStore
    - Data I/O (ROOT I/O, Corsika Input)
    - G4Svc for Geant4-based simulation
- Some applications have been moved to Lodestar:
    - KM2A fast simulation
    - WFCTA simulation
    - G4Argo: toy example for the G4-based simulation
- Results are compared to make sure the software runs correctly

Applications

- KM2A fast simulation (from LIU Ye)
  - Implemented with KM2ADetSimAlg and KM2ARecAlg
  - Some comparison of the results:



Applications

- WFCTA Simulation (from Ma Lingling)
  - Implemented with WFCTADetSimAlg
  - Some comparison of the results:

	Lodestar	**Origin**
Corex, corey	30857.314453, 21321.181641	30857.314453, 21321.181641
Zenith, azimuth	38.221312, 4.110079	38.221312, 4.110079
Total photon	720363989.469576	720363989.469576
Photons arriving telescope	2204600	2204600
Photon after reflecting	1432315	1432315
Photon after ray tracing	220824	220824

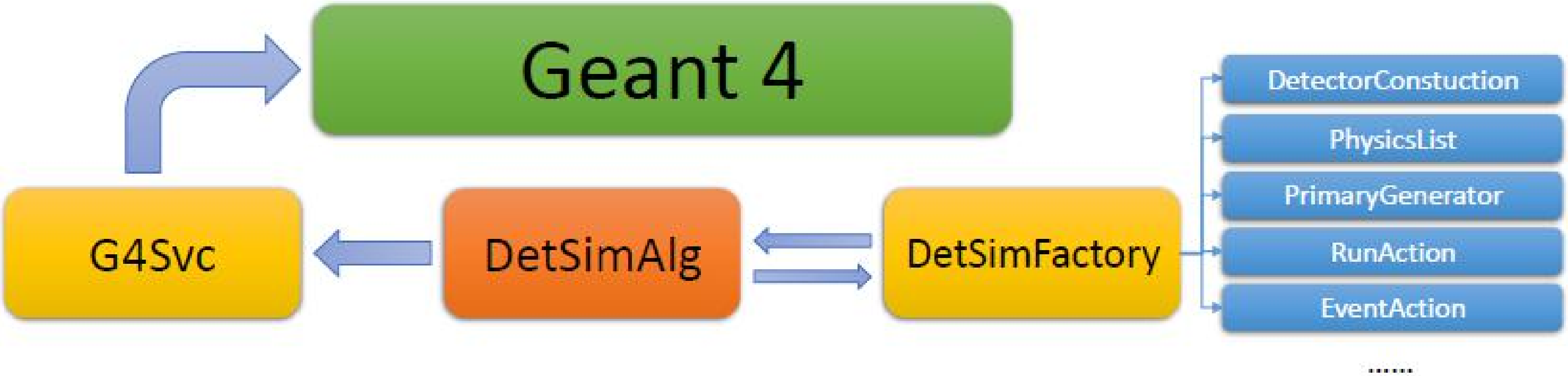
Applications

- To help developers to build the Geant4-based simulation software, a detsim framework is built in Lodestar:
- Components:
  - DetSimAlg: Common algorithm of Geant4 detector simulation
  - G4Svc: Interface to the Geant4 core
  - DetSimFactory: Build all the detector simulation options

    Generator

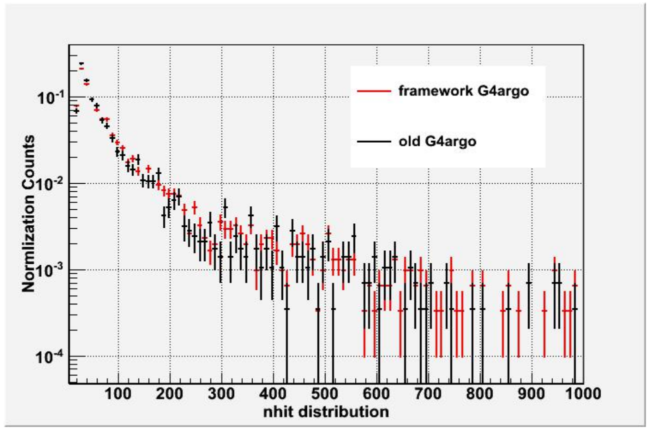
    Geometry

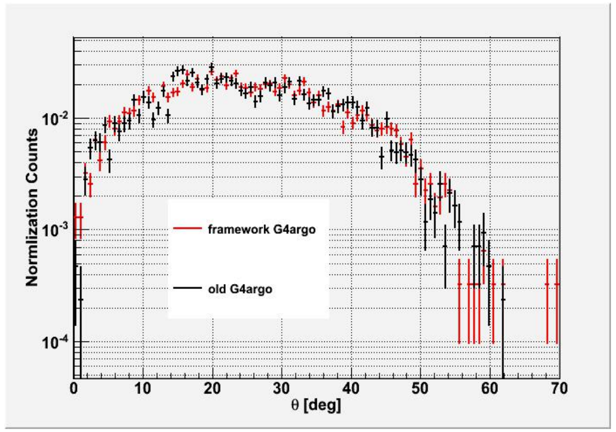
    User-actions

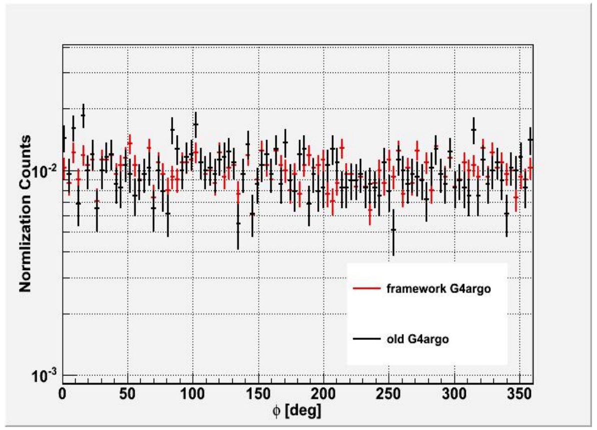


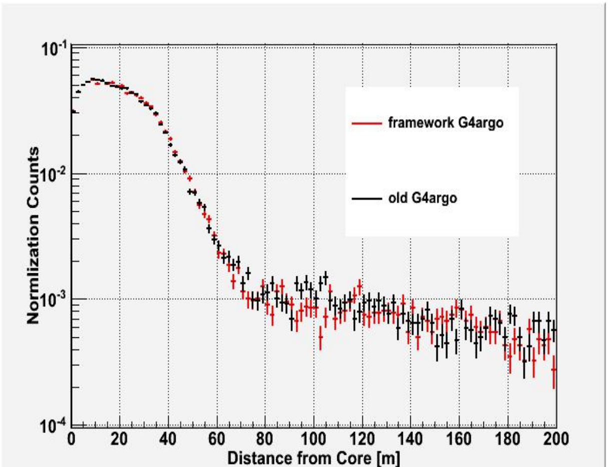
Applications

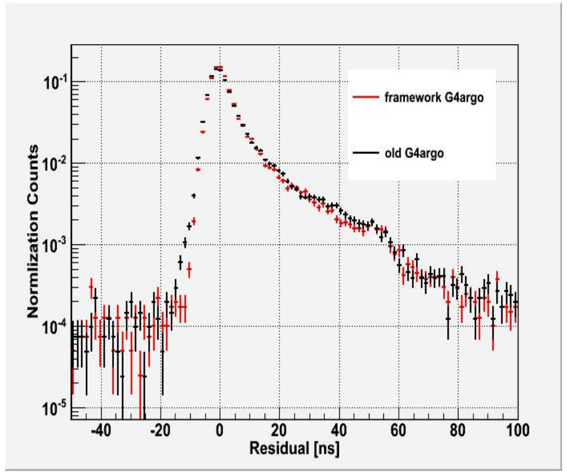
- G4Argo is provided as an example (From Guo Yiqing and Tian Zhen):
  - Use ArgoSimFactory to build all simulation options
  - Support input event splitting
  - Some comparison of the results:











Summary

- We have introduced:
  - Overview of the LHAASO Offline Software infrastructure and its functionalities
  - SNiPER framework
  - Several application examples moved into Lodestar
- Next to do:
  - Further improvements of Lodestar needs more considerations and discussions
  - Many tools and services to be added
  - Lots of existing packages to be moved to Lodestar
  - Build our SVN/Trac server
  - Lots of implementation work to do…
  - …