Progress of LHAASO Software Framework

Wenhao Huang, Xingtao Huang

Shandong University

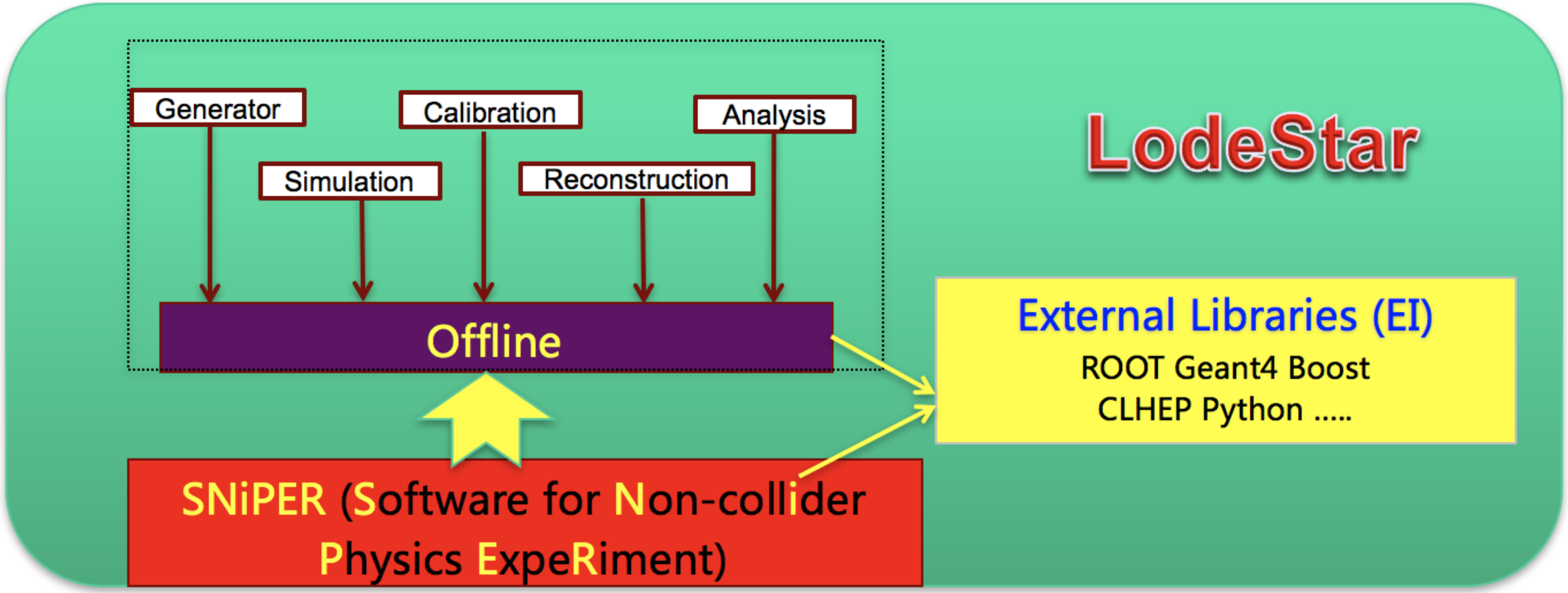
2019.04.13-15 Nanjing

Outline

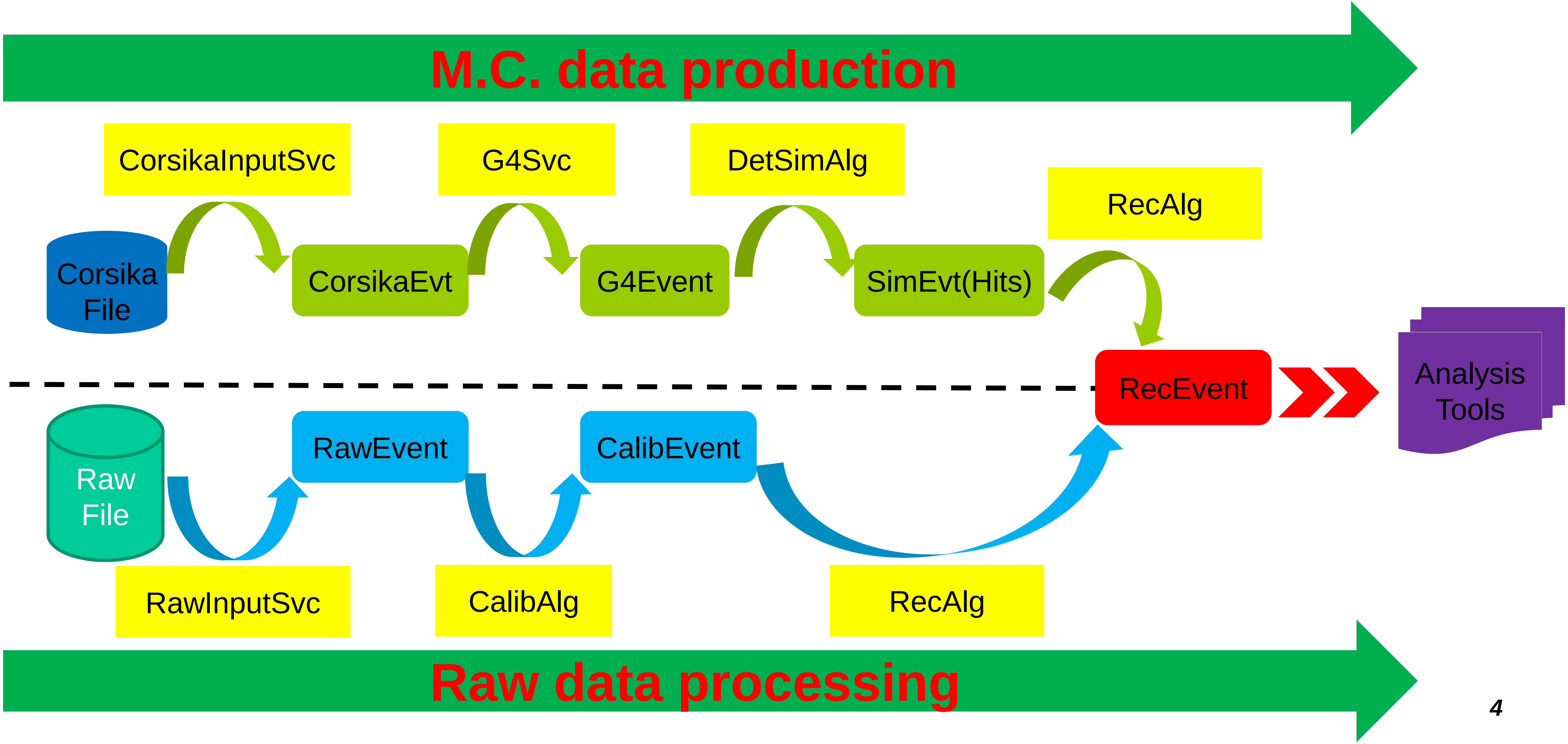
- ◆ Overview of LHAASO Offline Software System
- ◆ Optimization of Simulations
- ◆ Implementation of Reconstructions
- ◆ Current Status of Data Flow of raw data
- ◆ Summary & Plan

Overview of LHAASO Offline Software

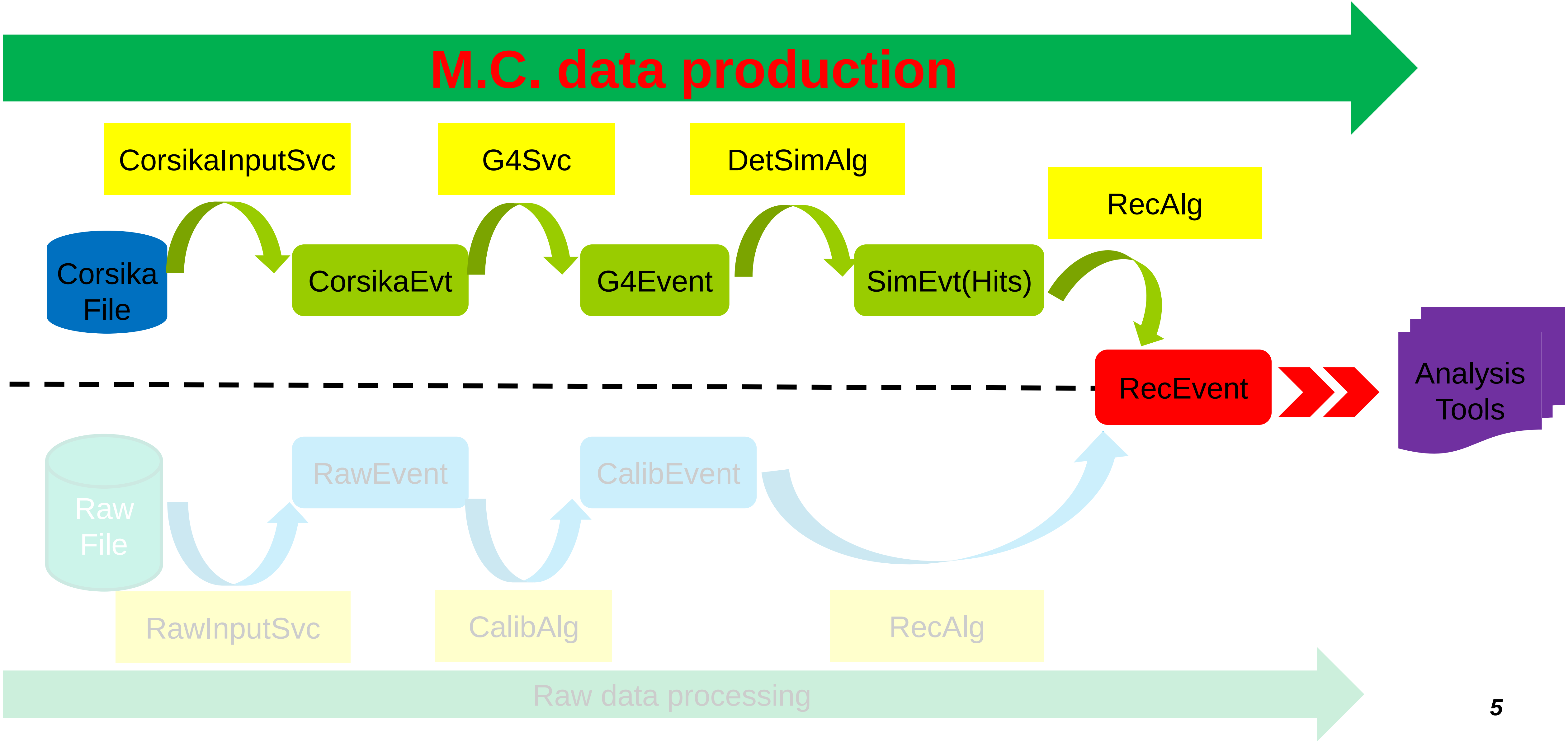
- LodeStar
  - **L**HAASO **O**ffline **D**ata Proc**e**ssing **S**of**t**ware Fr**a**mewo**r**k
- Main Components:
  - Offline: specific to LHAASO Experiments
  - SNiPER: underlying framework
  - External Libraries: frequently used third-party software or tools



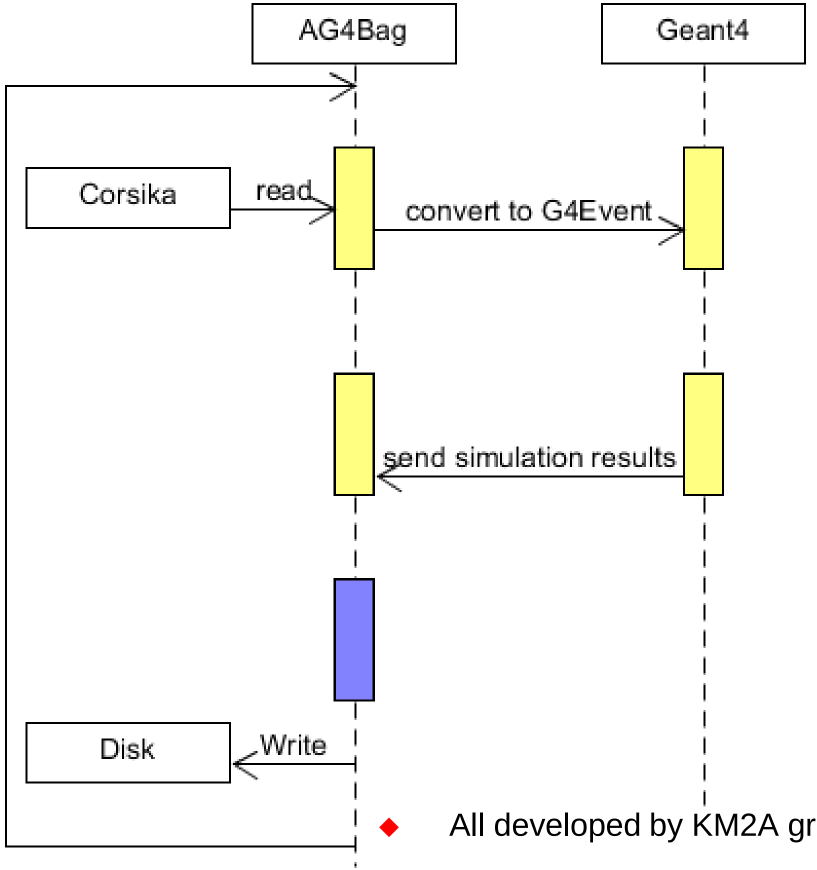
Two Data Flows in Framework



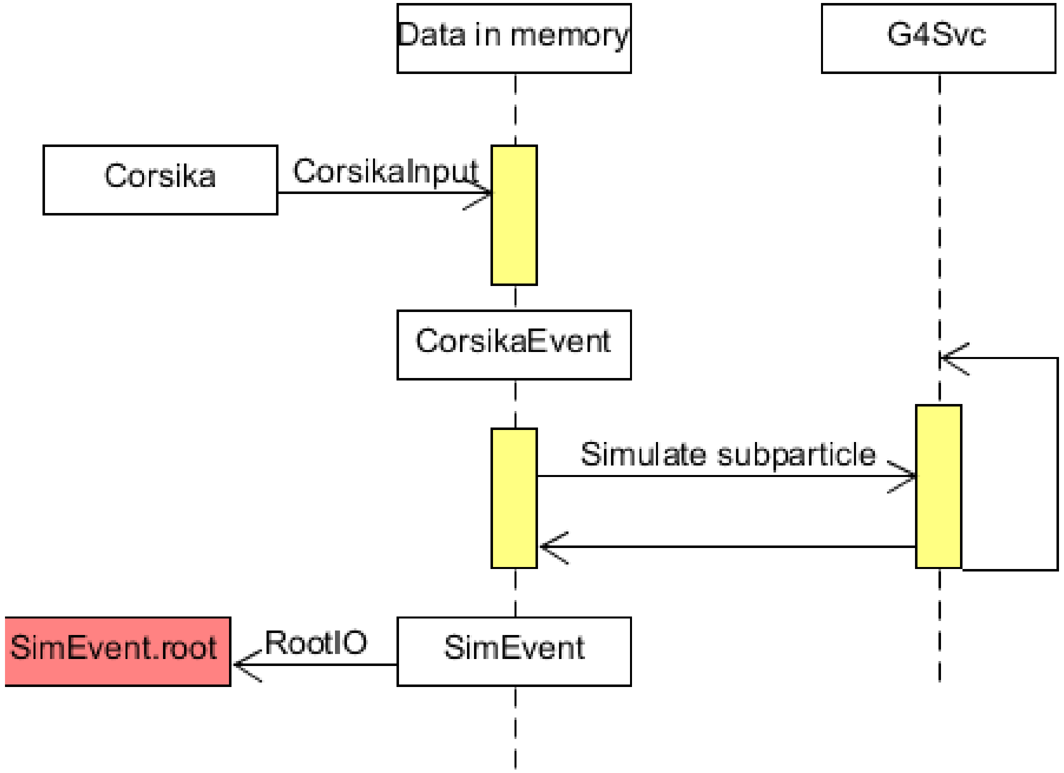
Two Data Flows in Framework



Optimizations in KM2A Simulation

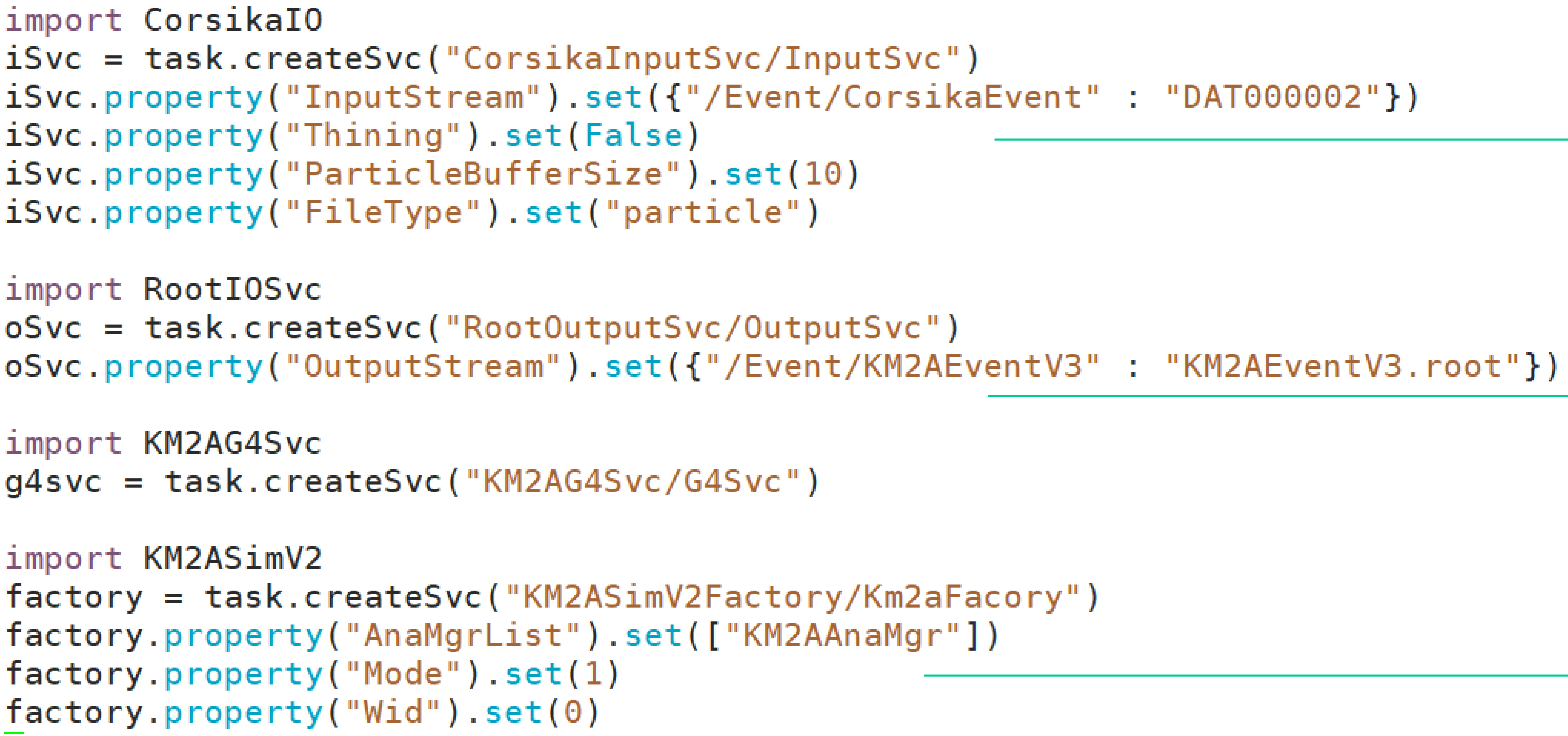


- All developed by KM2A group
  - Reading corsika
  - Event loop
  - Writing into file
- Codes are used theirself



- Implemented in framework way
  - Corsika input
  - G4Svc
  - RootIO
- Codes can be used in other applications

KM2A Simulation Script



```
import CorsikaIO
iSvc = task.createSvc("CorsikaInputSvc/InputSvc")
iSvc.property("InputStream").set({"/Event/CorsikaEvent" : "DAT000002"})
iSvc.property("Thining").set(False)
iSvc.property("ParticleBufferSize").set(10)
iSvc.property("FileType").set("particle")

import RootIOSvc
oSvc = task.createSvc("RootOutputSvc/OutputSvc")
oSvc.property("OutputStream").set({"/Event/KM2AEventV3" : "KM2AEventV3.root"})

import KM2AG4Svc
g4svc = task.createSvc("KM2AG4Svc/G4Svc")

import KM2ASimV2
factory = task.createSvc("KM2ASimV2Factory/Km2aFacory")
factory.property("AnaMgrList").set(["KM2AAnaMgr"])
factory.property("Mode").set(1)
factory.property("Wid").set(0)
```

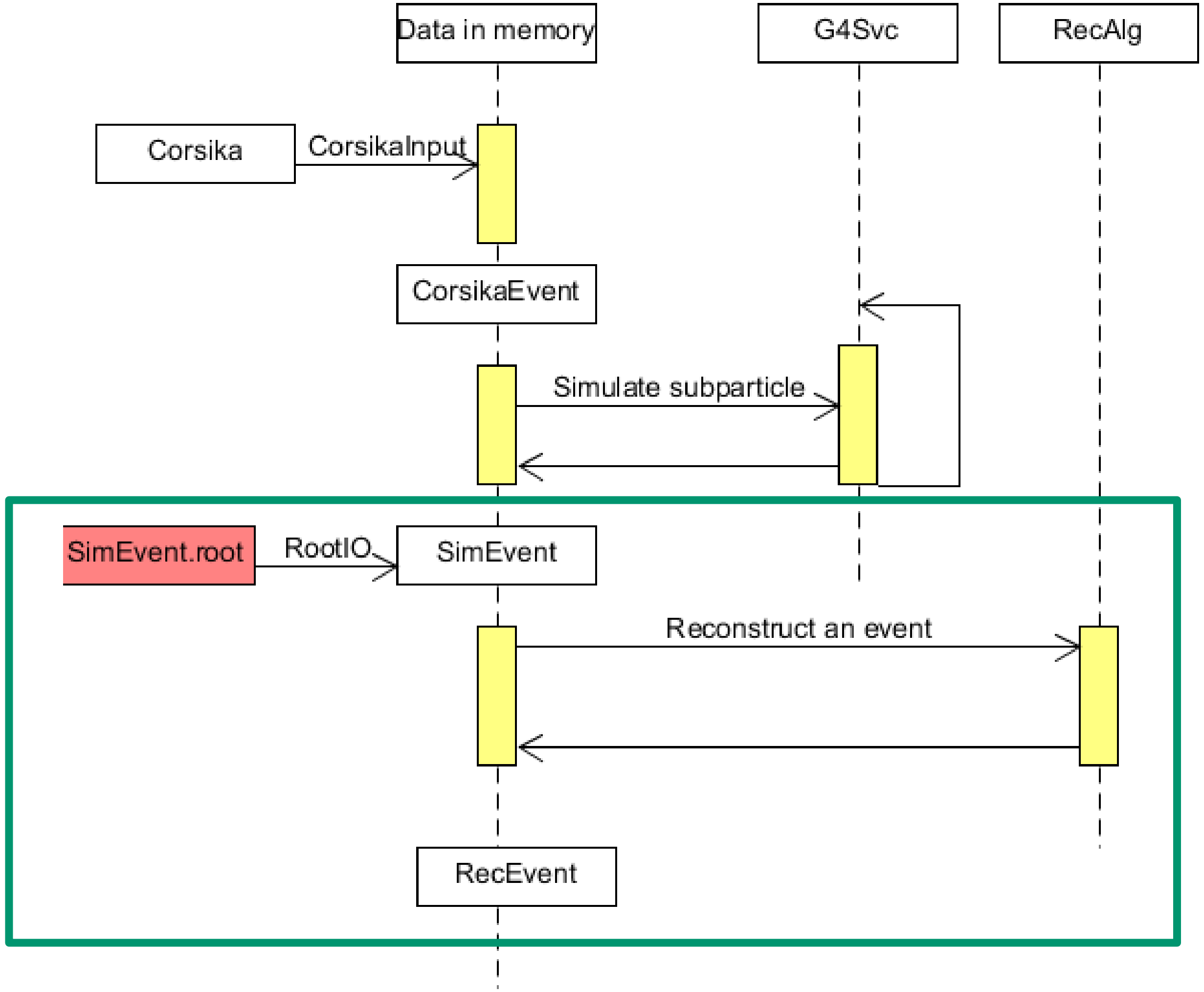
Input Service

Output Service

Simulation Parameters

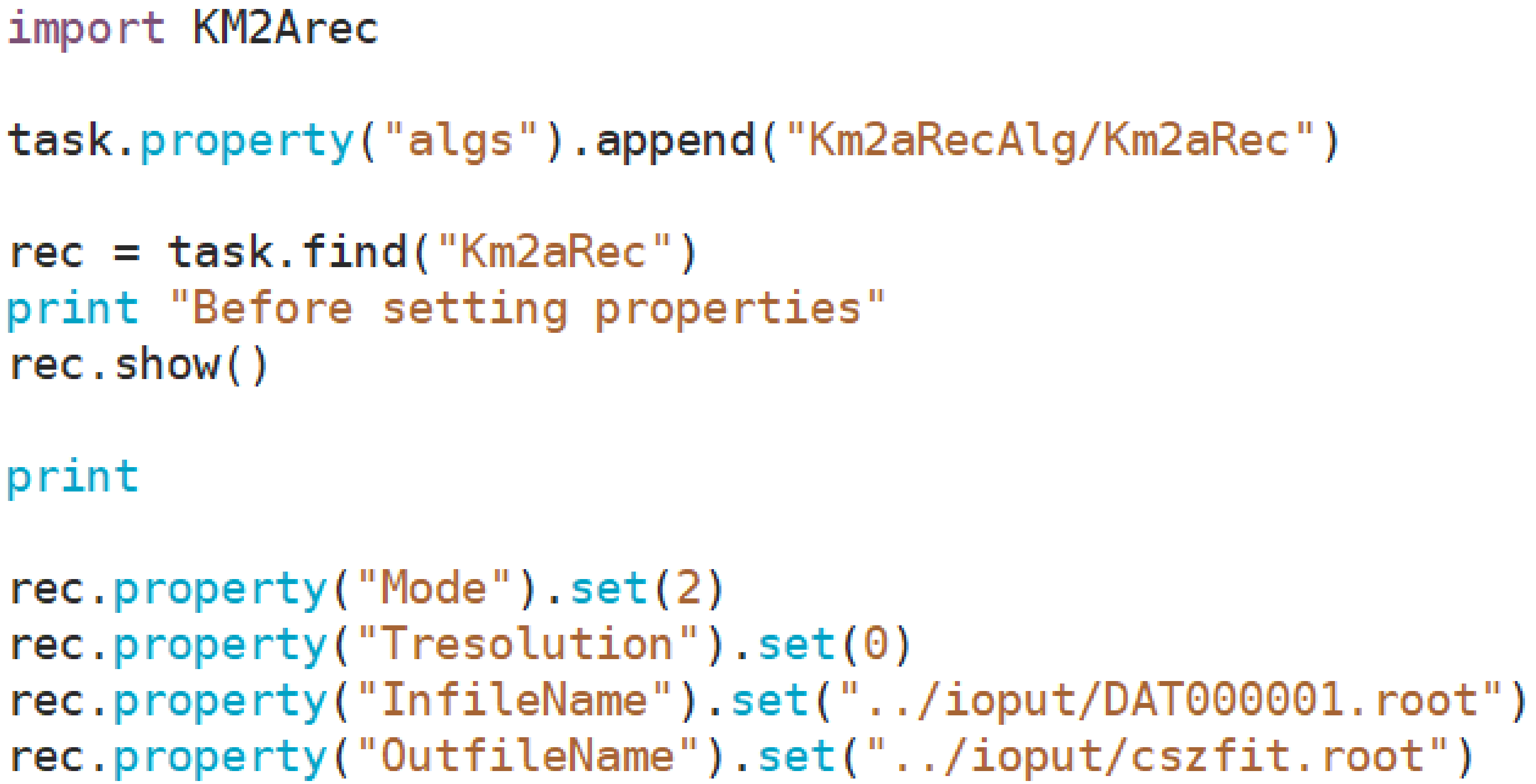
$python DetSim_KM2A.py

KM2A(Reconstruction)



- Easily adding the reconstruction algorithms
  - Direction reconstruction
  - core position reconstruction

KM2A(Reconstruct of direction and core position)



```
import KM2Arec

task.property("algs").append("Km2aRecAlg/Km2aRec")

rec = task.find("Km2aRec")
print "Before setting properties"
rec.show()

print

rec.property("Mode").set(2)
rec.property("Tresolution").set(0)
rec.property("InfileName").set("../ioput/DAT000001.root")
rec.property("OutfileName").set("../ioput/cszfit.root")
```

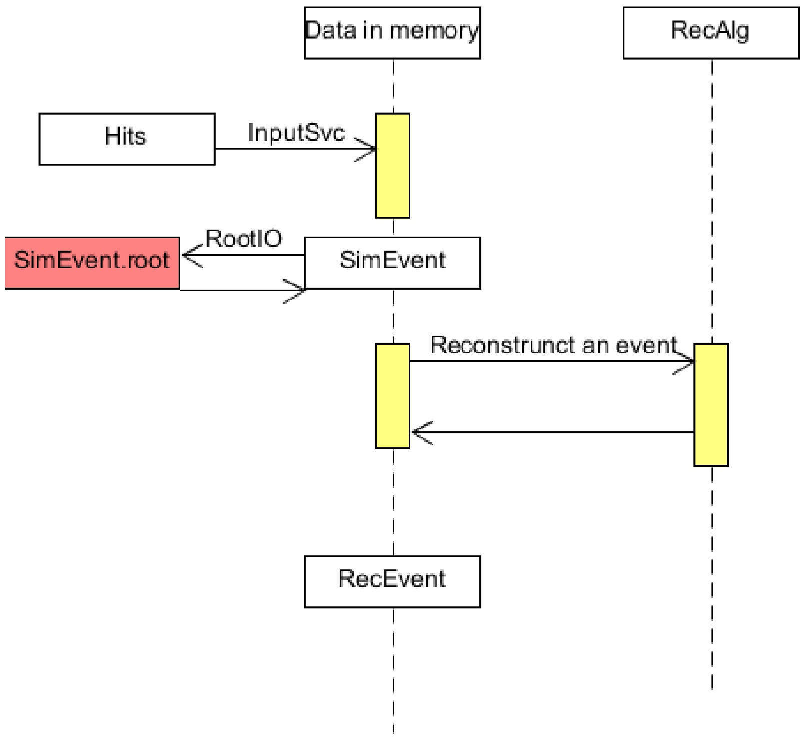
$python Rec_KM2A.py

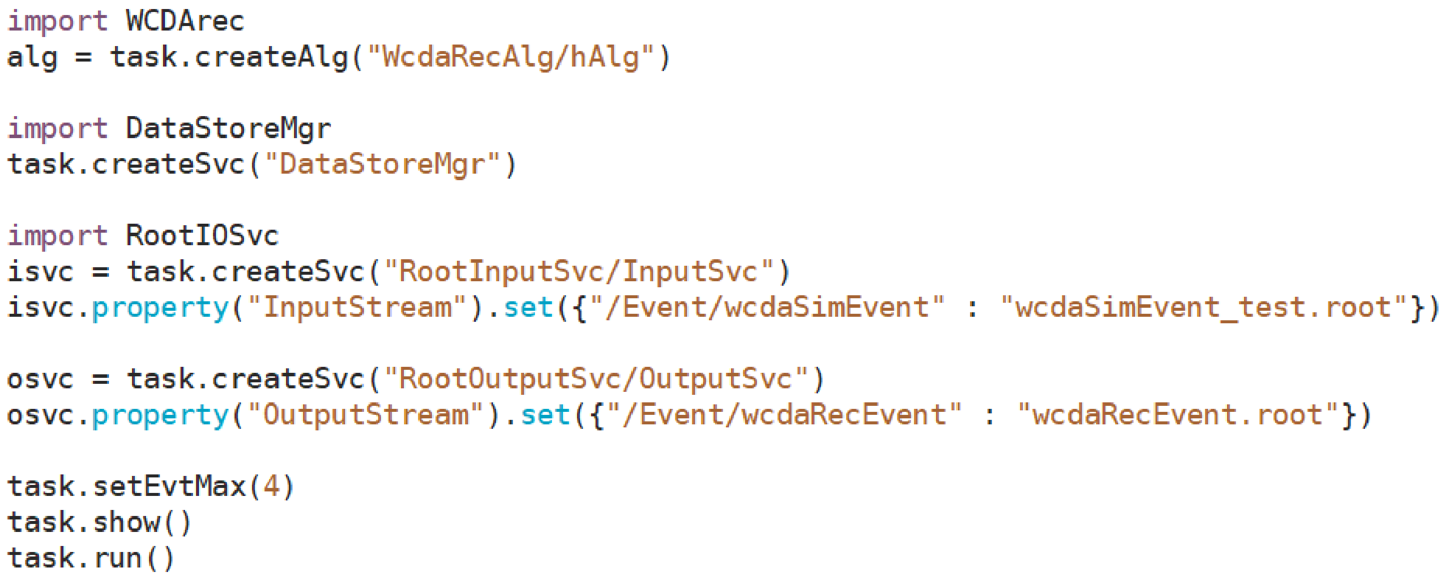
Several lines in the **job configurations**, the **reconstruction algorithms** will be **automatically** called and write the reconstructed results into the root files

WCDA(Simulation & Reconstruction)

$python DetSim_WCDA_1(2).py inputfile outputfile –settingfile …

$python wcdaHitsConv.py





```
import WCDArec
alg = task.createAlg("WcdaRecAlg/hAlg")

import DataStoreMgr
task.createSvc("DataStoreMgr")

import RootIOSvc
isvc = task.createSvc("RootInputSvc/InputSvc")
isvc.property("InputStream").set({"/Event/wcdaSimEvent" : "wcdaSimEvent_test.root"})

osvc = task.createSvc("RootOutputSvc/OutputSvc")
osvc.property("OutputStream").set({"/Event/wcdaRecEvent" : "wcdaRecEvent.root"})

task.setEvtMax(4)
task.show()
task.run()
```

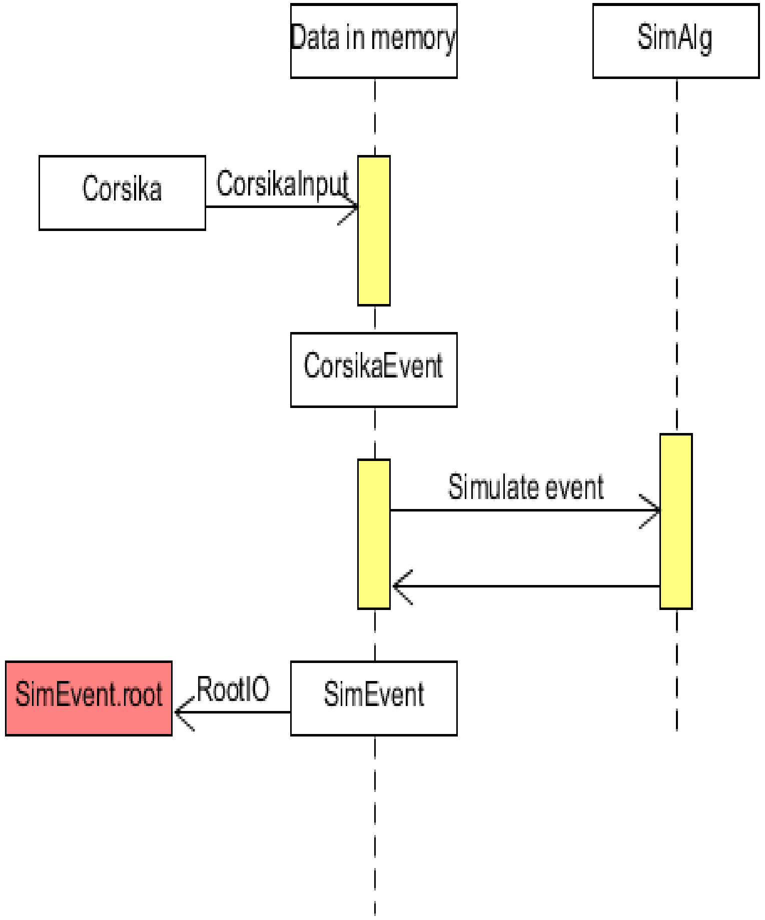
$python Rec_WCDA.py

- Framework reads simulated hits into the data store
  - Write them into the root files as KM2A
  - Also can be used by the Reconstruction algorithms

- WCDA simulation can be run with the unified way.
  - No much changes on the WCDA detector simulation

WFCTA Simulation

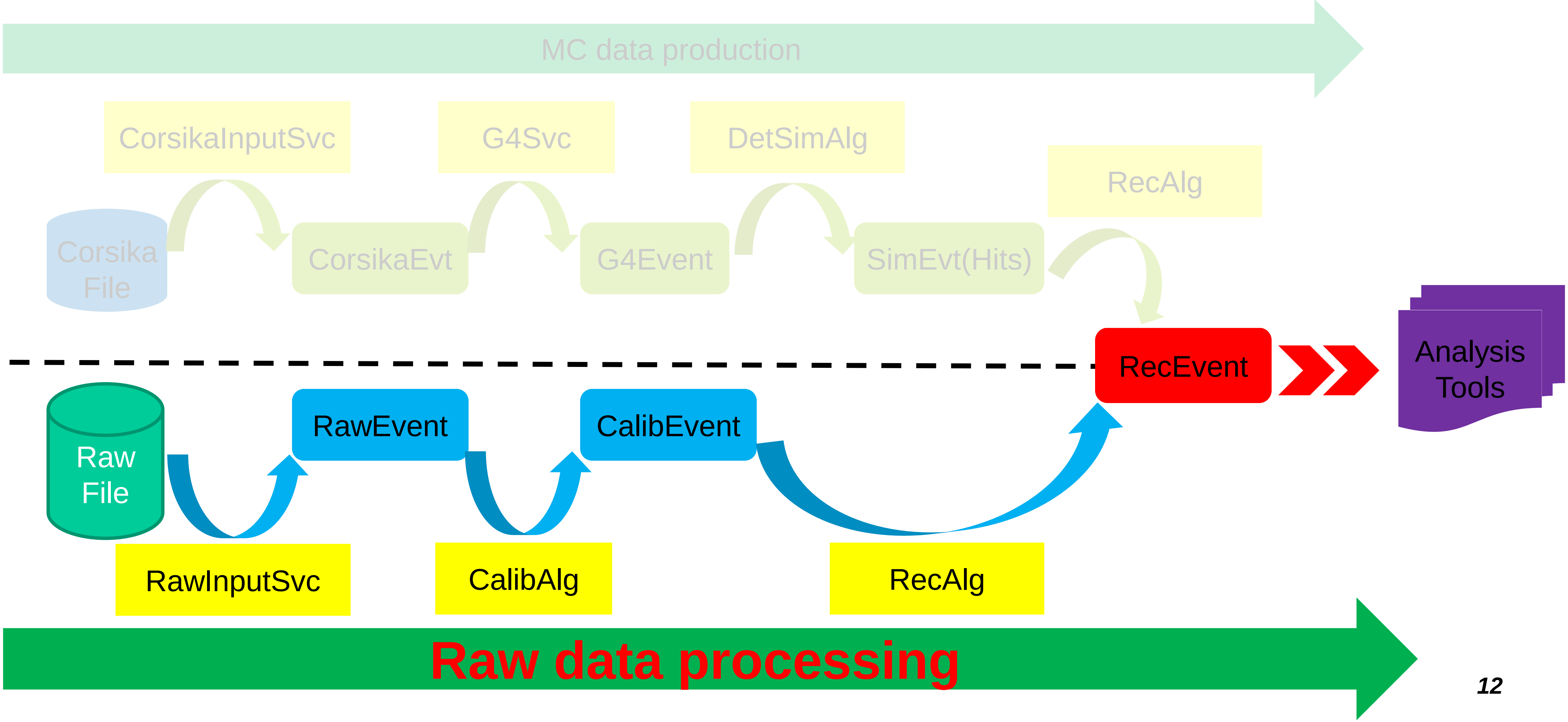
- We updated data model of WFCTA(WFCTASimEvent).



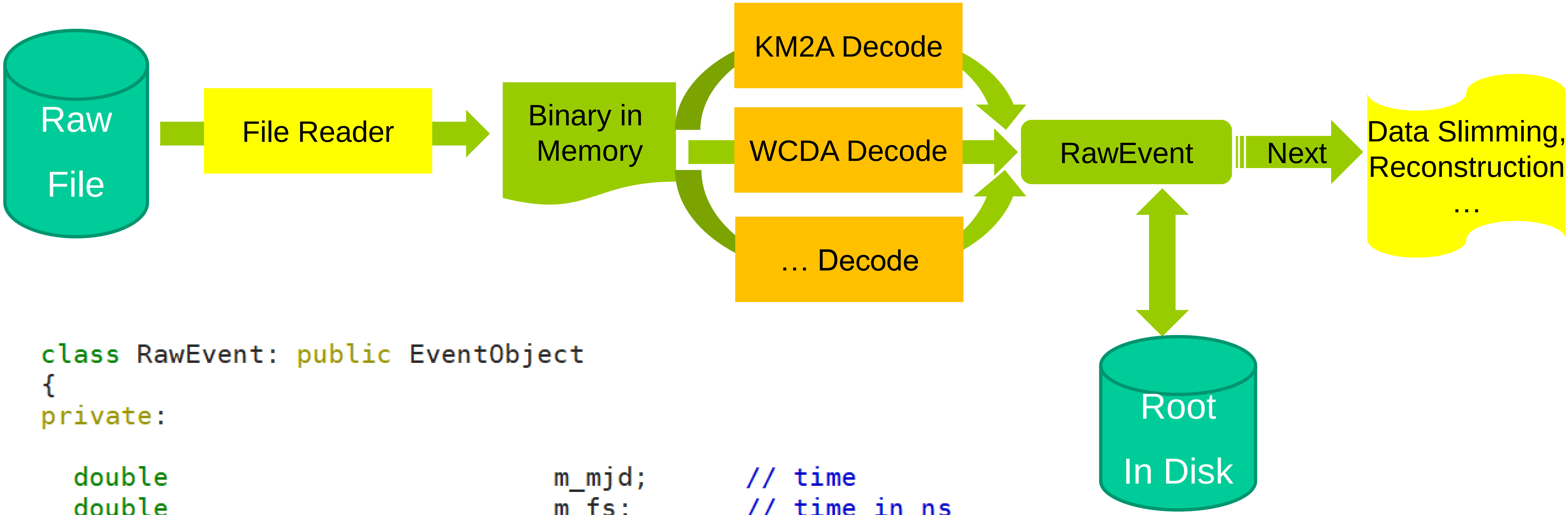
```
import WFCTASim
detalg = Task.createAlg("WFCTADetSimAlg/det_sim_alg")
detalg.property("optionfile").set("default.inp")

import DataStoreMgr
Task.createSvc("DataStoreMgr")

import CorsikaIO
iSvc = Task.createSvc("CorsikaInputSvc/InputSvc")
iSvc.property("InputStream").set({"/Event/CorsikaEvent" : "/afs/ihep.ac.cn/users/l/llma/run01/CER000029"})
iSvc.property("Thining").set(True)
iSvc.property("ParticleBufferSize").set(10000)
iSvc.property("FileType").set("cherenkov")

import RootIOSvc
oSvc = Task.createSvc("RootOutputSvc/OutputSvc")
oSvc.property("OutputStream").set({"/Event/WFCTASimEvent" : "SimEvent.root"})
```

```
$python DetSim_WFCTA.py
```

Two Data Flows in Framework



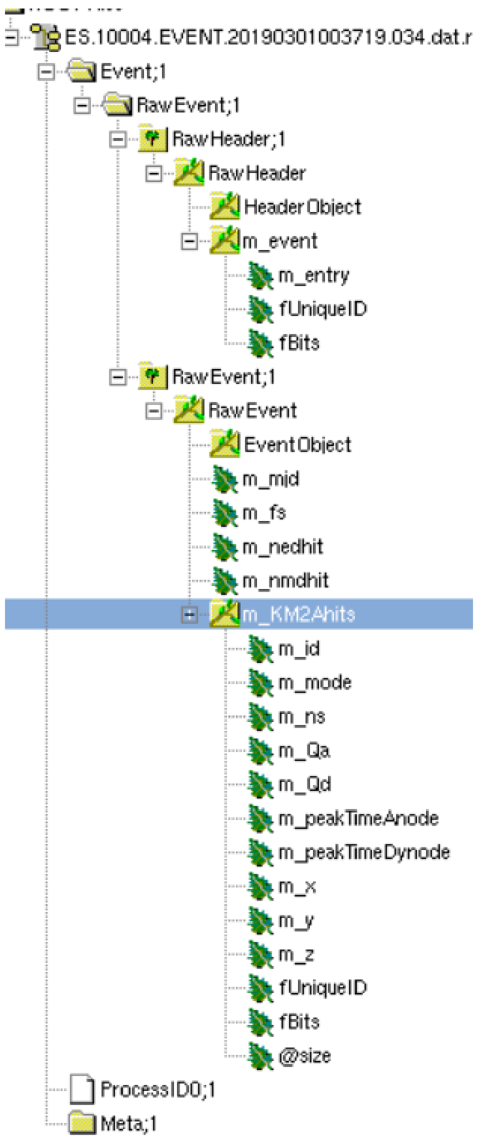
Raw Data Input Module



```
class RawEvent: public EventObject
{
private:

   double                      m_mjd;      // time
   double                      m_fs;       // time in ns
   int                         m_nedhit;   // number of ed hits
   int                         m_nmdhit;   // number of md hits
   std::vector<LHAASO::KM2AHit*> m_KM2Ahits; // List of KM2A hits
```

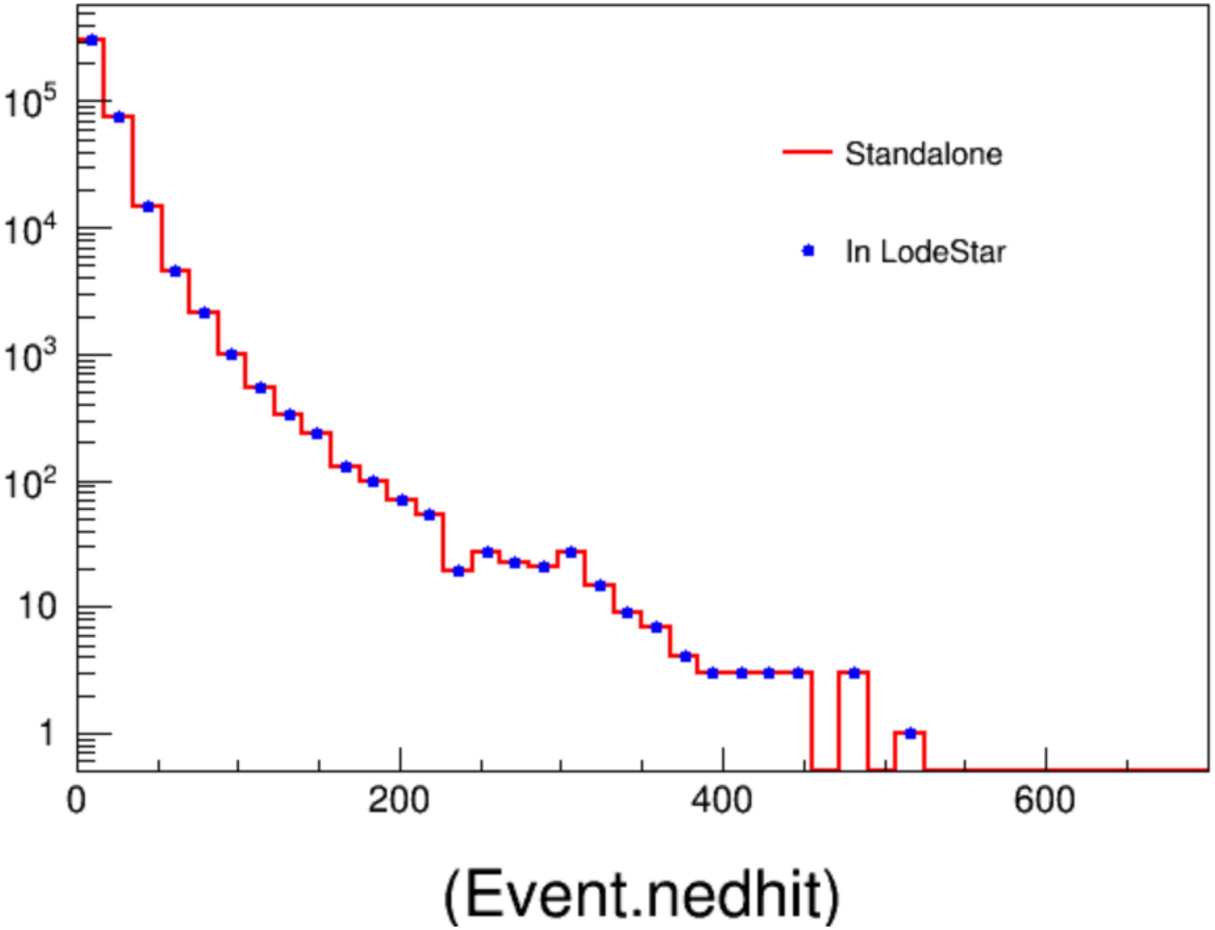
Raw Data Input Module

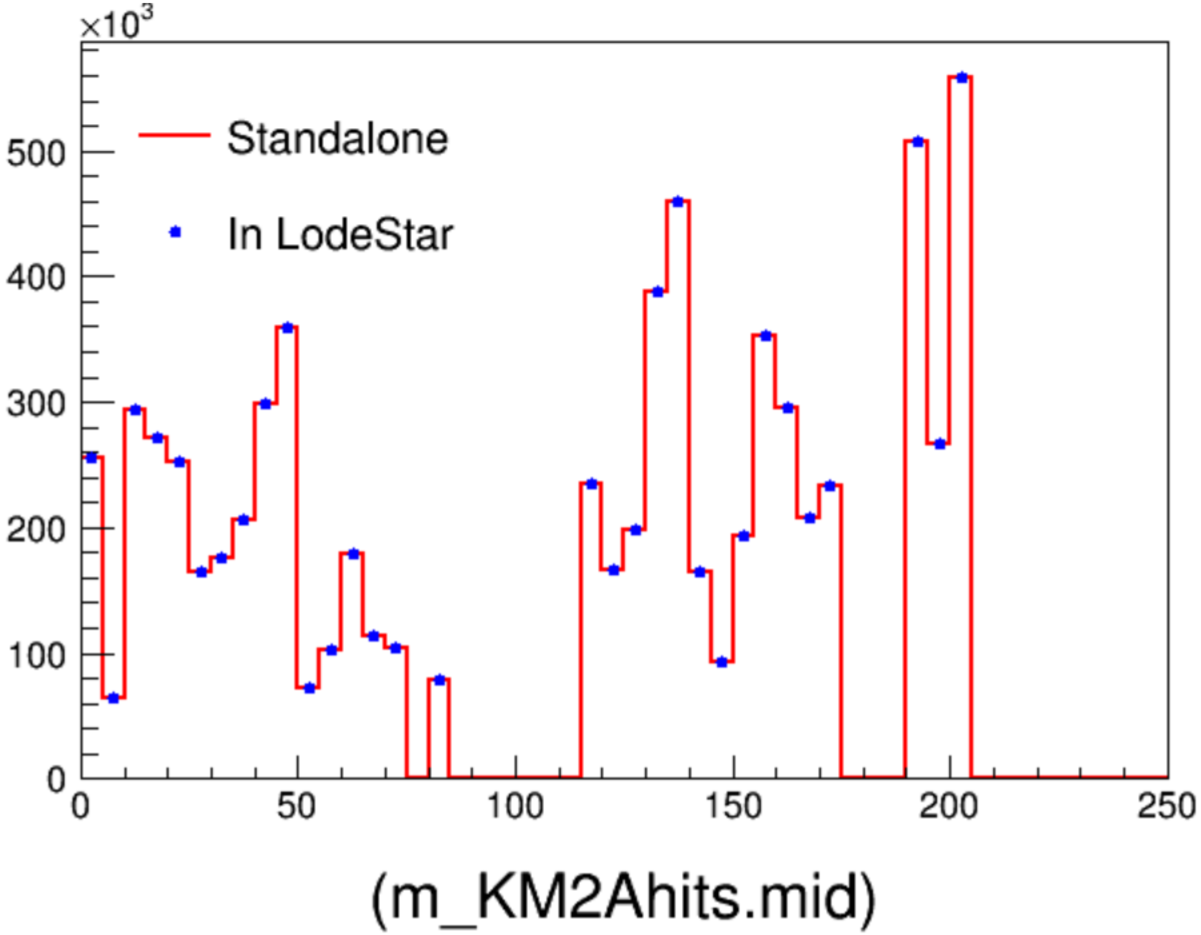
```
#! /usr/bin/env python
import Sniper

Algtask = Sniper.Task("Algtask")

import IOTestAlg
datalg = Algtask.createAlg("IOTestAlg/myalg")

import RawIO
iSvc = Algtask.createSvc("RawInputSvc/InputSvc")
iSvc.property("InputStream").set({"/Event/RawEvent" : "/scratchfs/ybj/zhanghy/LodeStar/E
S.10003.EVENT.20190226015650.013.dat"})


import DataStoreMgr
Algtask.createSvc("DataStoreMgr")

import RootIOSvc
oSvc = Algtask.createSvc("RootOutputSvc/OutputSvc")
oSvc.property("OutputStream").set({"/Event/RawEvent" : "test.root"})

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



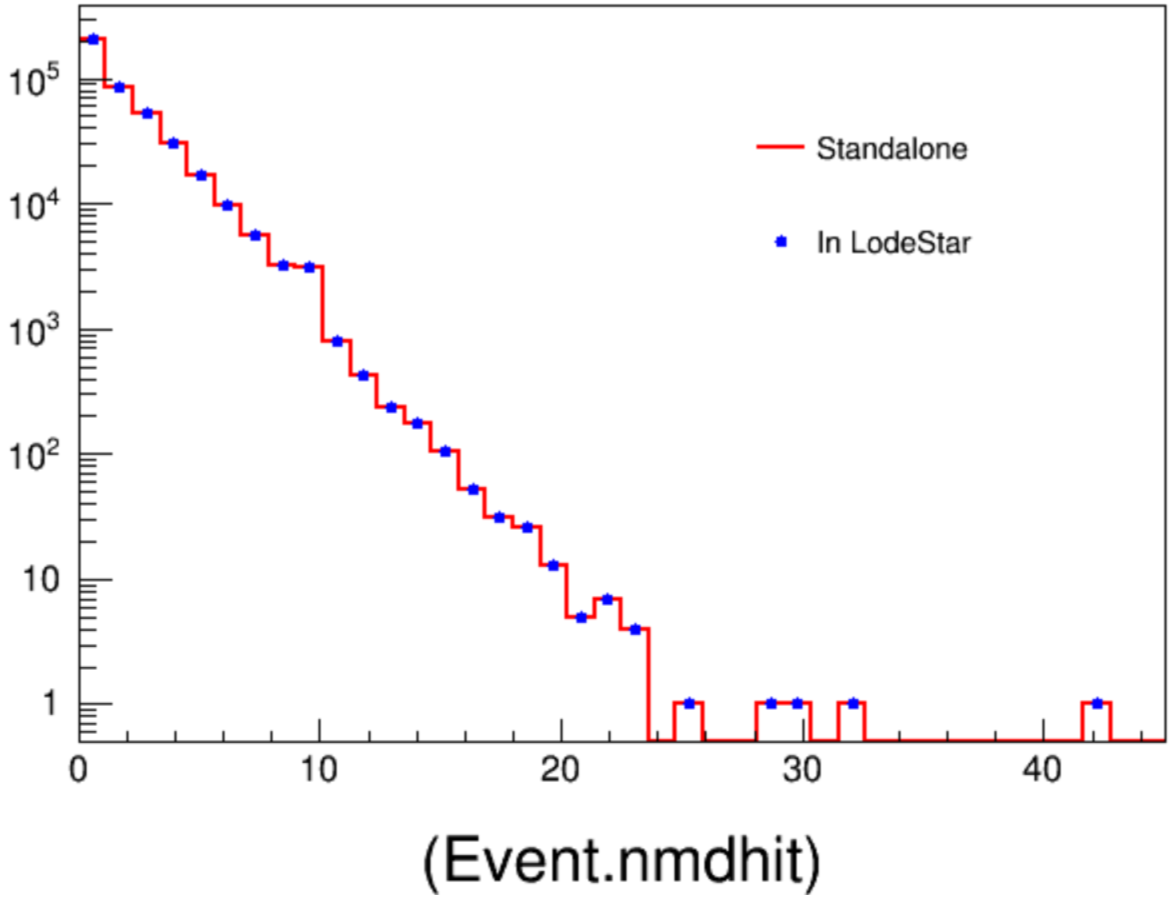
```
ES.10004.EVENT.20190301003719.034.dat.r
  Event;1
    RawEvent;1
      RawHeader;1
        RawHeader
          HeaderObject
          m_event
            m_entry
            fUniqueID
            fBits
      RawEvent;1
        RawEvent
          EventObject
          m_mjd
          m_fs
          m_nedhit
          m_nmdhit
          m_KM2Ahits
            m_id
            m_mode
            m_ns
            m_Qa
            m_Qd
            m_peakTimeAnode
            m_peakTimeDynode
            m_x
            m_y
            m_z
            fUniqueID
            fBits
            @size
  ProcessID0;1
  Meta;1
```

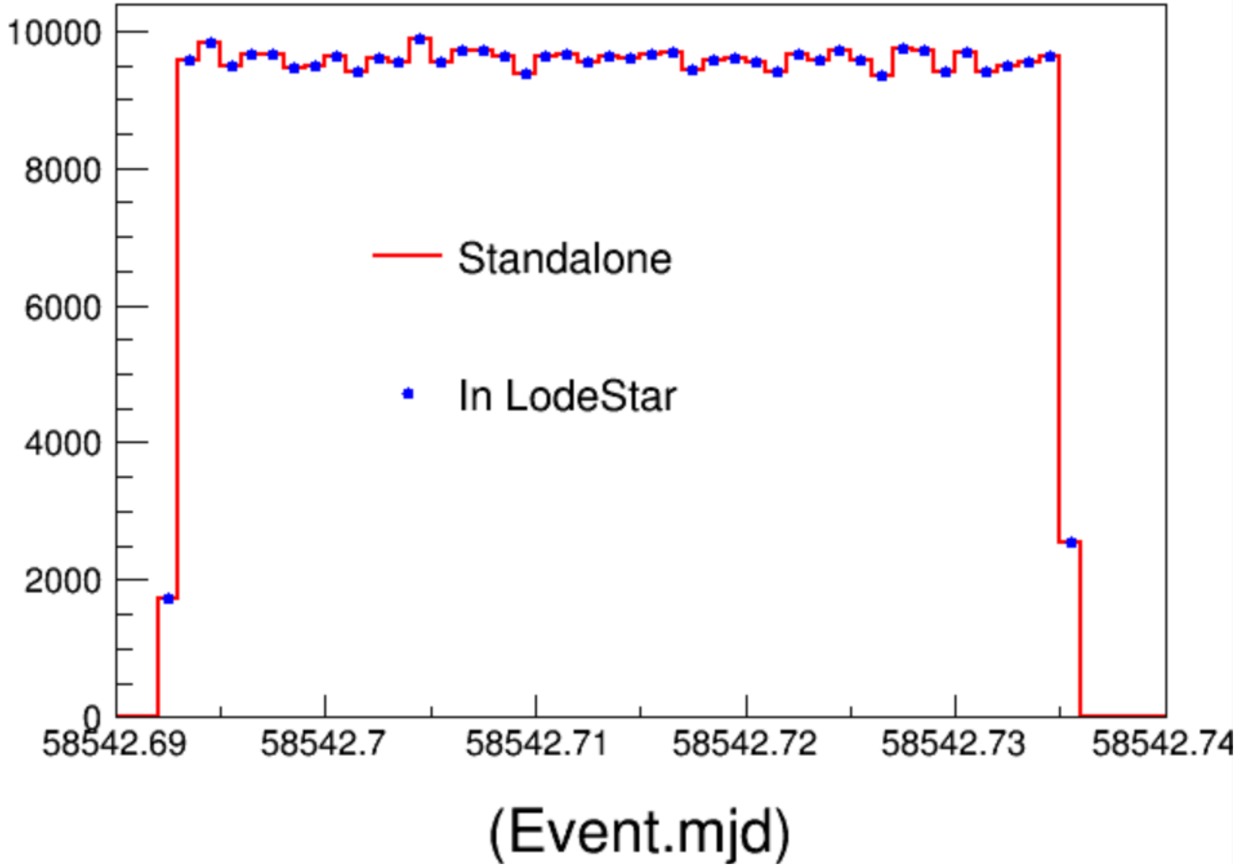
Raw Data Results Validation



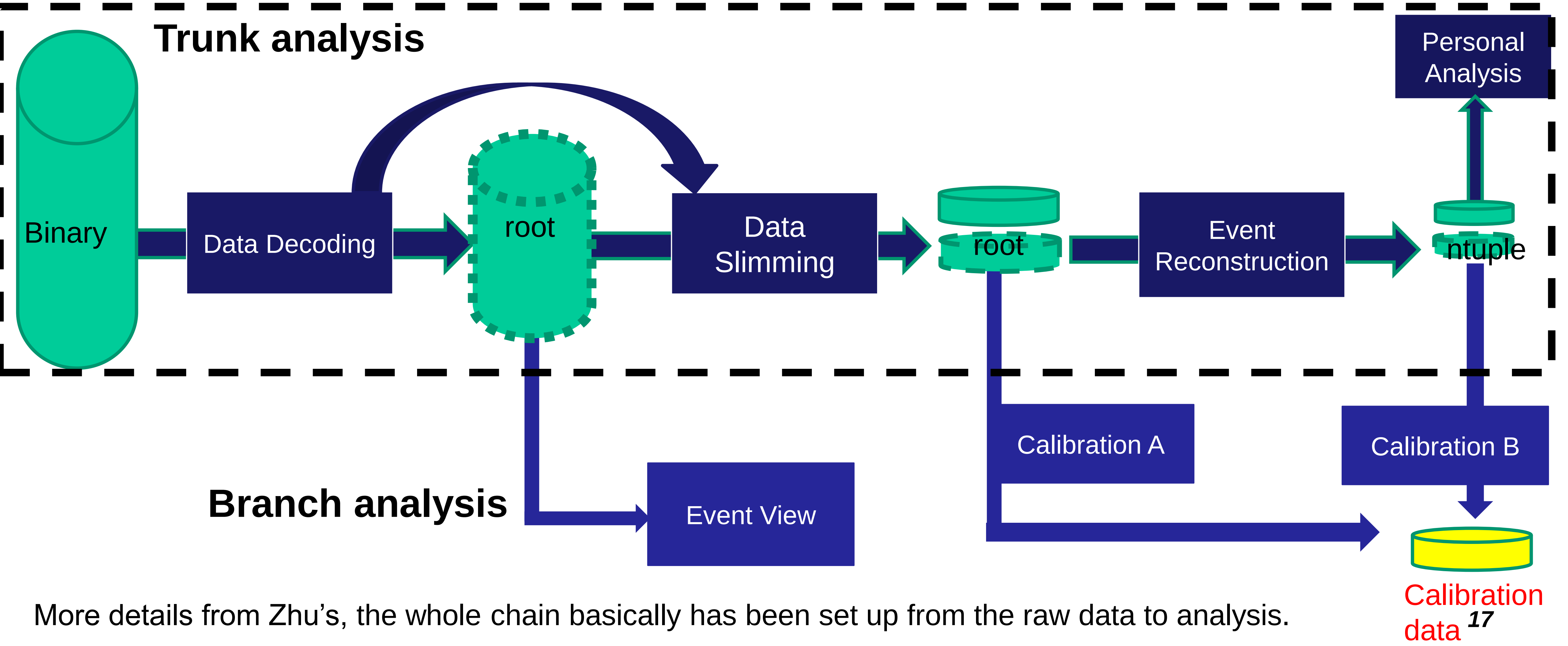


Raw Data Results Validation





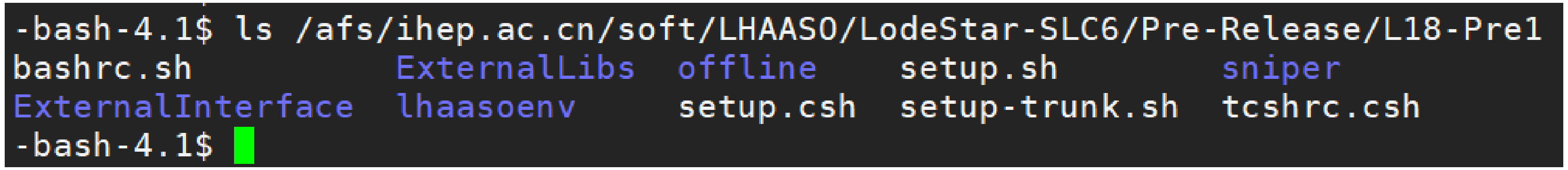
Data Flow of Raw Data



More details from Zhu's, the whole chain basically has been set up from the raw data to analysis.

Latest version and documentation

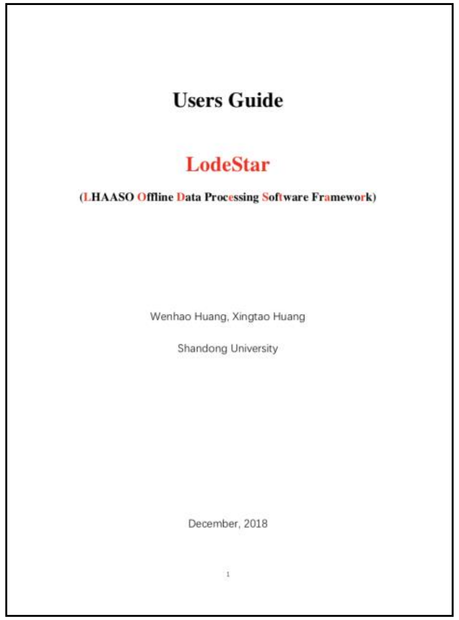
- The latest version of LodeStar has been installed at ihep.
  - /afs/ihep.ac.cn/soft/LHAASO/LodeStar-SLC6/Pre-Release/L18-Pre1



```
-bash-4.1$ ls /afs/ihep.ac.cn/soft/LHAASO/LodeStar-SLC6/Pre-Release/L18-Pre1
bashrc.sh            ExternalLibs   offline     setup.sh         sniper
ExternalInterface    lhaasoenv      setup.csh   setup-trunk.sh   tcshrc.csh
-bash-4.1$
```

- **Doc-Db**

  581-v3



Users Guide

LodeStar

(LHAASO Offline Data Processing Software Framework)

Wenhao Huang, Xingtao Huang

Shandong University

December, 2018

1

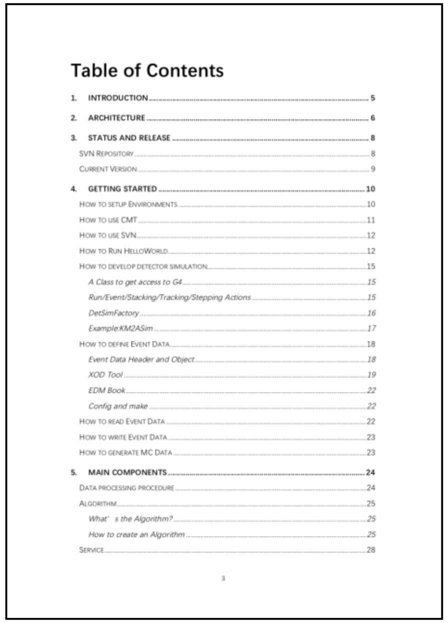
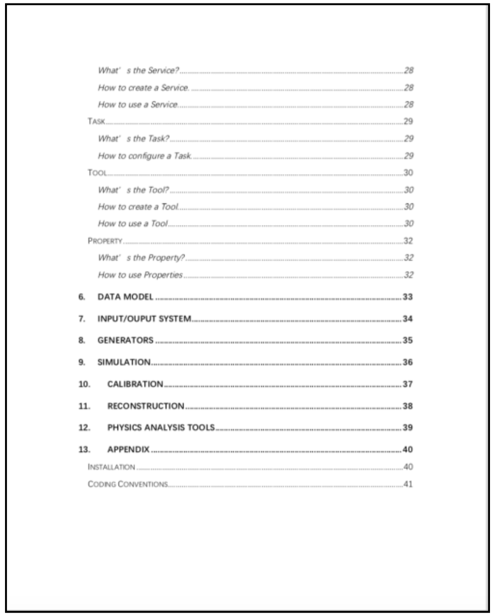


Table of Contents

3



Summary & Plan

- Optimized data flow in framework
  - M.C. Data
  - Raw Data
- KM2A, WCDA and WFCTA have been integrated with LodeStar
  - More people and imputes are needed
- Raw Data Input Service has been integrated with LodeStar

- Develop calibration and event select algorithm.
- Setup the whole chain from Corsica (Raw) data to physics analysis

Thank you

Advantage

- Modularity
  - common modules
  - specific modules
  - share
- Independent
  - Low coupling between different modules
- Easy to access
  - Modules are easy to combine in python scripts.