Progress on integration with LodeStar

Wenhao Huang, Haohao Wang, Na Yin, Xingtao Huang

Shandong University

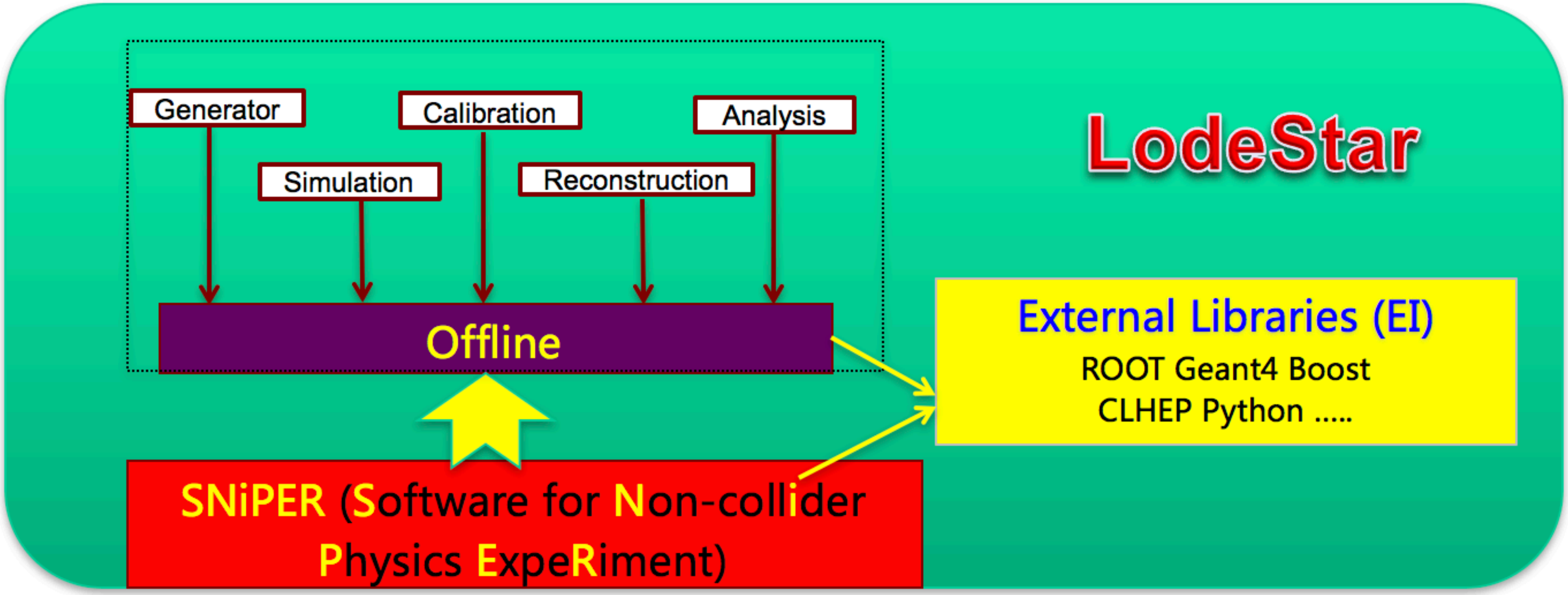
2018.10.10 LinZhi

Outline

- ◆ Overview of LHAASO Offline Software System
- ◆ Optimized Data Flow of Offline Data Processing
- ◆ Detector Simulation under LoadStar
- ◆ Event Data Model
- ◆ Input/Output System
- ◆ Some Examples
- ◆ Summary
- ◆ Planning

Overview of LHAASO Offline Software

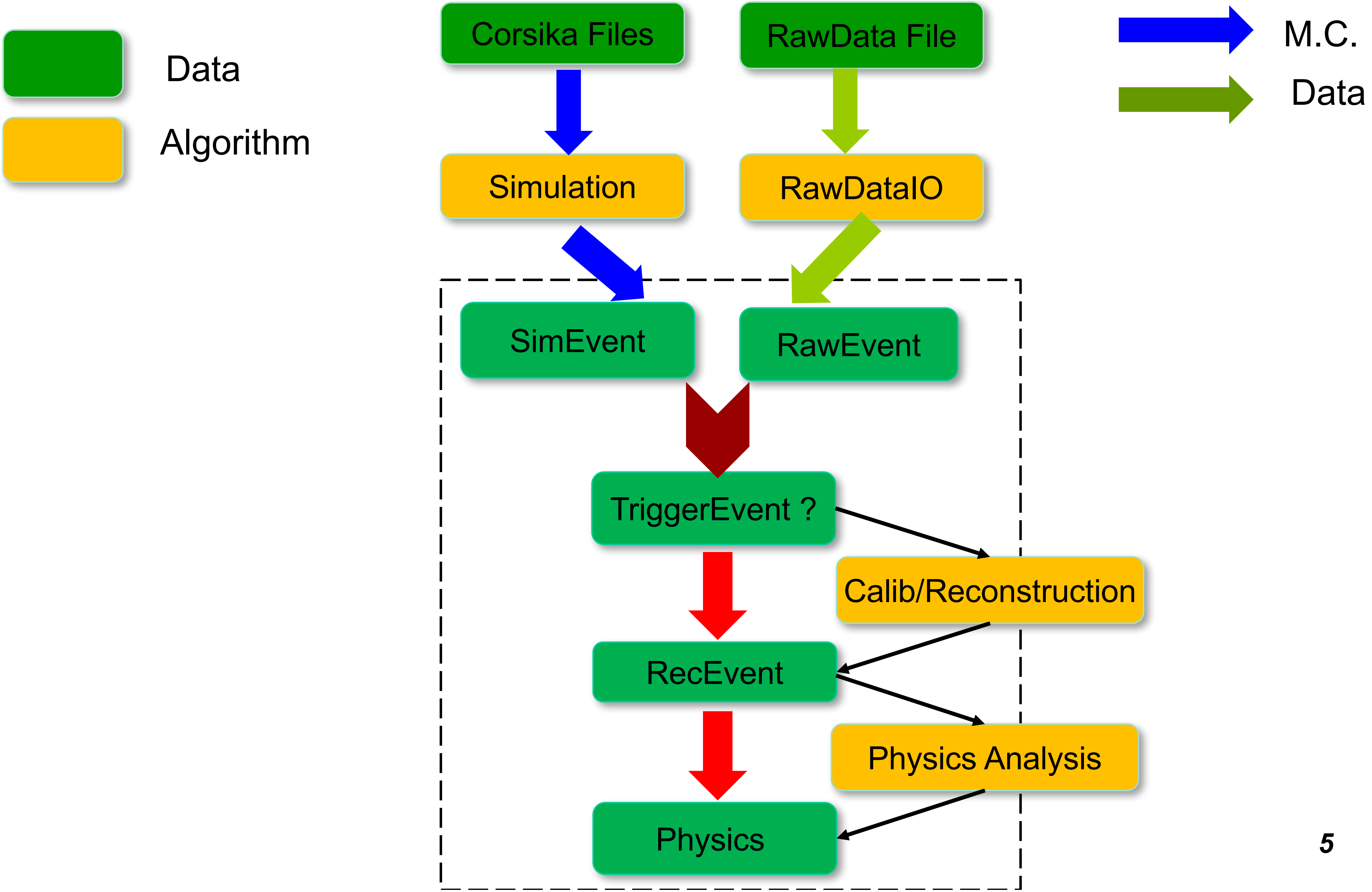
- LodeStar(北极星):
  - LHAASO Offline Data Processing Software Framework
- Main Components:
  - Offline: specific to LHAASO Experiments
  - SNiPER: underlying framework
  - External Libraries: frequently used third-party software or tools



Computing and Development Environments

- Supported Operator System: SCL
  - Scientific Linux 6 are currently supported
- Programming Language: C++,Python
  - very popular in HEP
  - most frequently used software implemented in C++
- Configuration Language： Python
  - Very flexible
  - configure jobs without re-compiling software
- Software Management Tool： CMT
  - Automatically compile, build packages
  - Automatically deal with relationships between package
- Version Control Tool： SVN
  - keep history of codes developing
  - synchronization and sharing between developers

Optimized Data Flow of Offline Data Processing



🟩 Data

🟨 Algorithm

M.C.

Data

Corsika Files

RawData File

Simulation

RawDataIO

SimEvent

RawEvent

TriggerEvent ?

Calib/Reconstruction

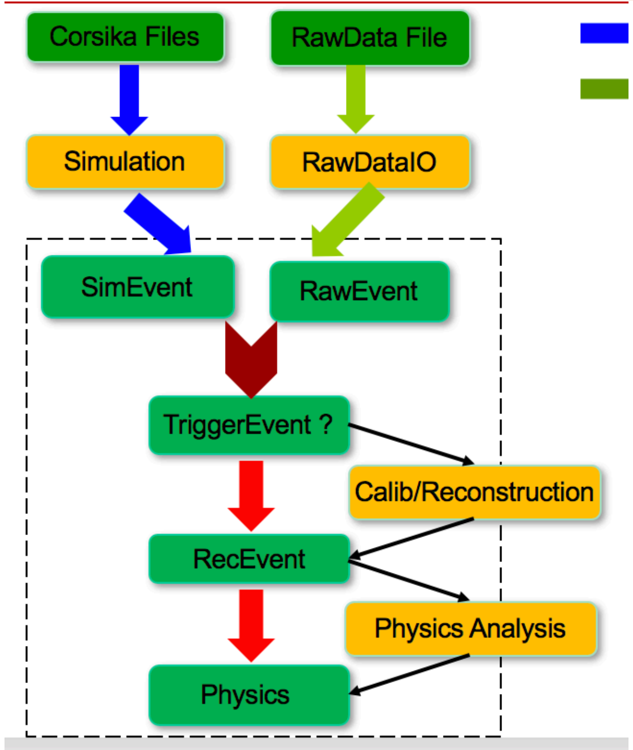
RecEvent

Physics Analysis

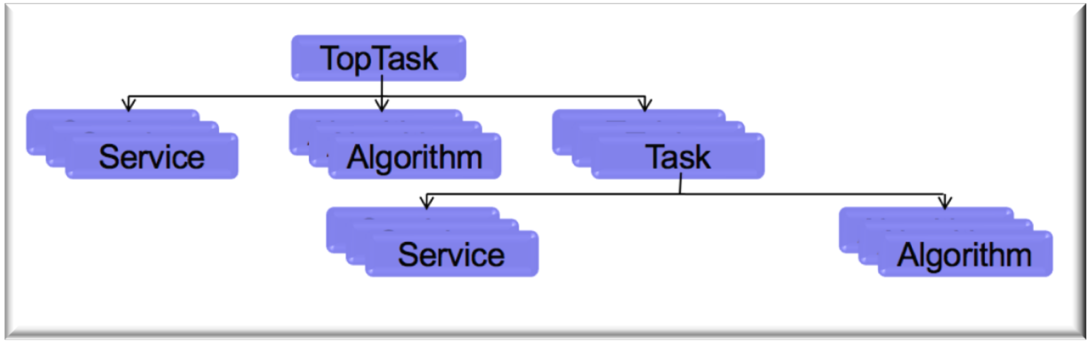
Physics

Functionalities provided by Framework

- From Users point of view:
  - Algorithm: **Yellow Box**
  - EventData: **Green Box**
  - Data Access Service: **Black Line**



- From Framework Developer point of view:
  - How to manage algorithms
  - How to manage services
  - How to manage event data
  - How to configure jobs



Algorithm

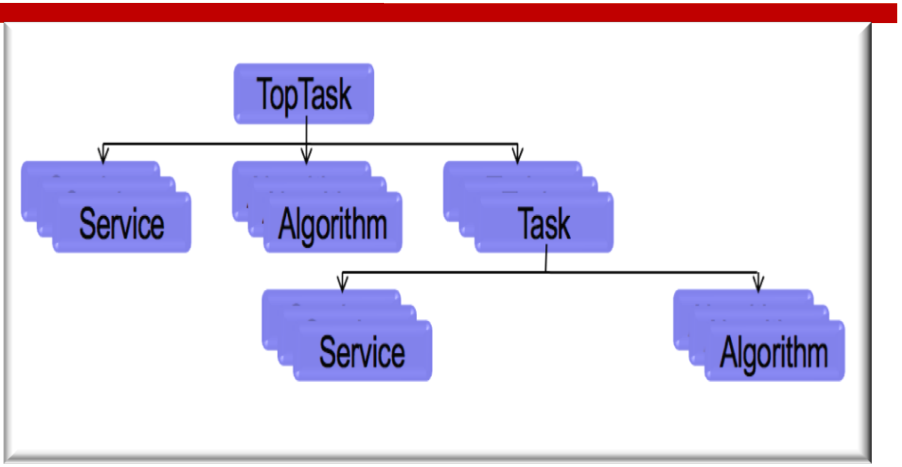
- An unit of codes for Data Processing
  - the calculation during event loop
  - Most frequently used by users

- AlgBase, the abstract base class provided by Framework
  - User’s algorithm must be inherited from AlgBase
  - Its constructor takes one std::string parameter

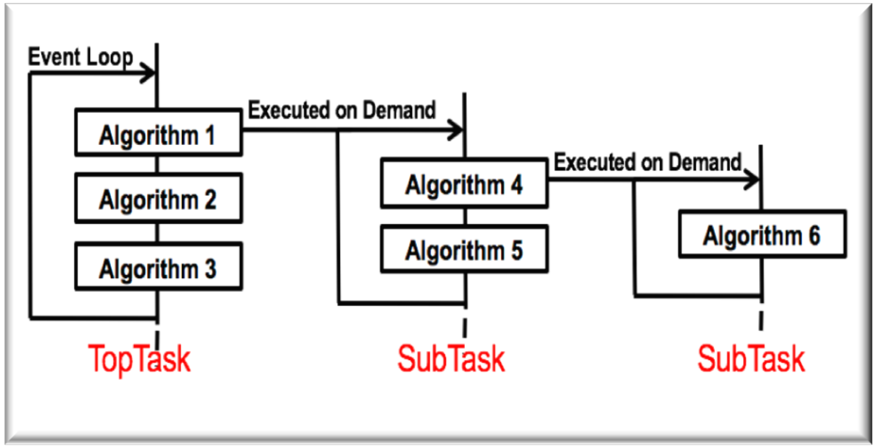
- 3 abstract functions must be implemented, which are called by SNiPER automatically
  - bool initialize() : called once per Task (at the beginning of a Task)
  - bool execute() : called once per Event
  - bool finalize() : called once per Task (at the end of Task)

Task

- A lightweight Application Manager
  - Management of algorithms, services and tasks
  - Controlling the execution of algorithms
  - Has its own data memory management
  - Has its own I/O management

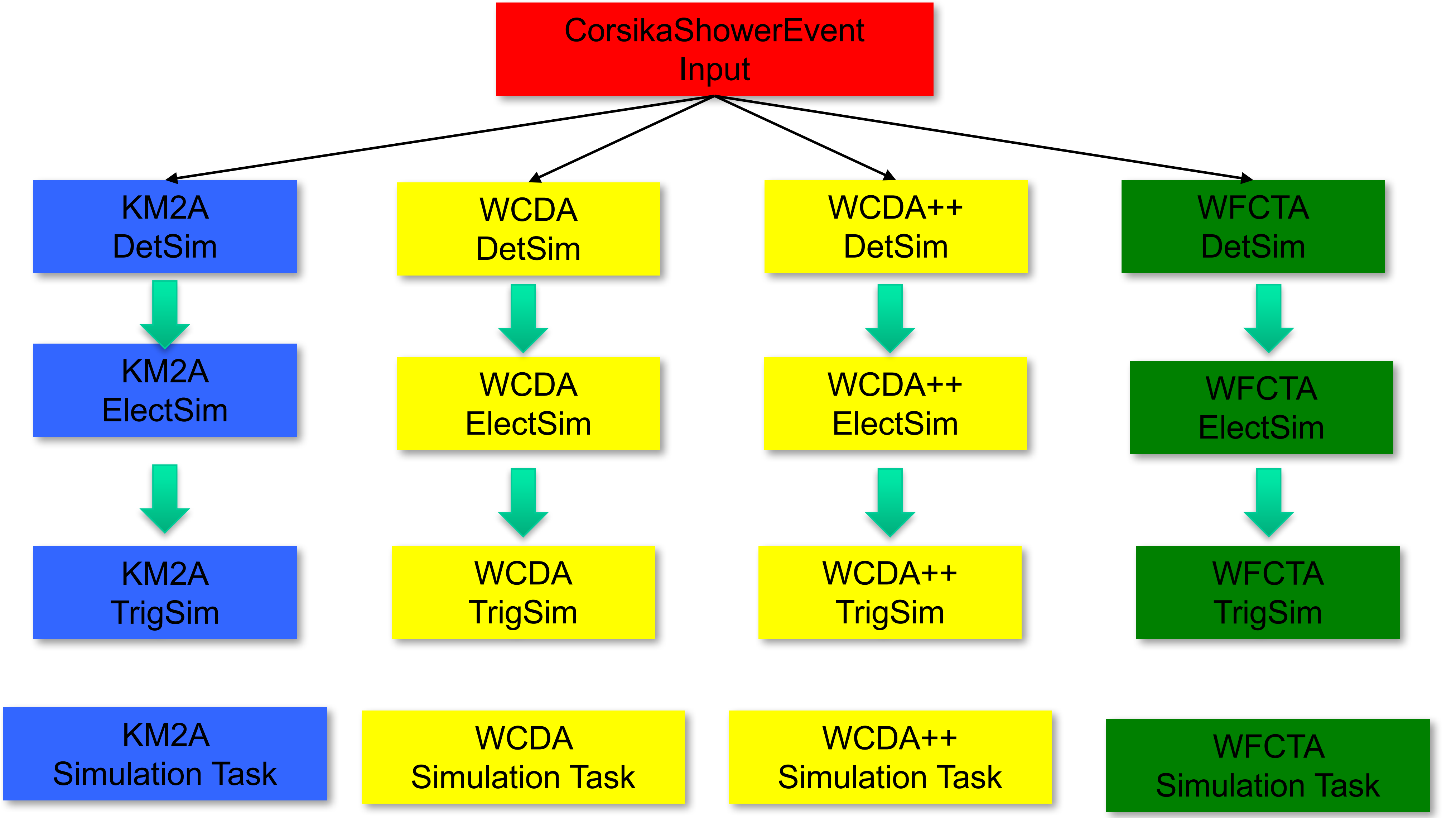


- One job can has more than one Tasks (e.g. KA2MSim, WFCTASim)



- Task and SubTask provide more flexible event execution

Detector Simulation Chain



CorsikaShowerEvent
Input

KM2A
DetSim

WCDA
DetSim

WCDA++
DetSim

WFCTA
DetSim

KM2A
ElectSim

WCDA
ElectSim

WCDA++
ElectSim

WFCTA
ElectSim

KM2A
TrigSim

WCDA
TrigSim

WCDA++
TrigSim

WFCTA
TrigSim

KM2A
Simulation Task

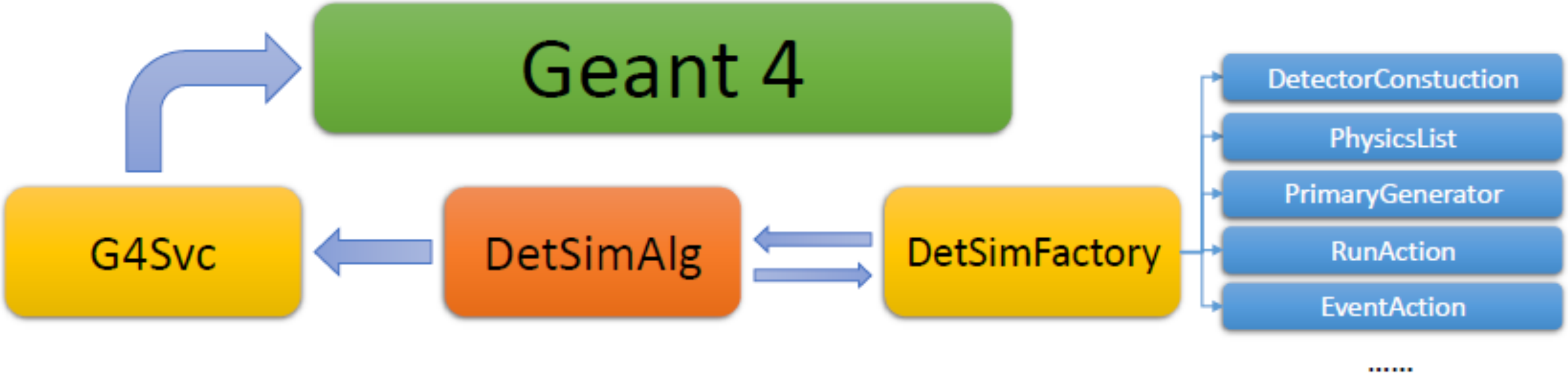
WCDA
Simulation Task

WCDA++
Simulation Task

WFCTA
Simulation Task

Detector Simulation

- LodeStar manages detector simulation with **Task**， which consists of Algorithms and services
- A dedicated algorithm (**DetSimAlg**) for all sub-detectors simulation
- A dedicated service (**G4Svc**) for launching Geant4 within LodeStar
- A user-end service(**DetSimFactory**) for set up and organize all the Geant4 related classes, such as
  - G4VUserDetectorConstruction
  - G4VUserPhysicsList
  - G4VUserPrimaryGeneratorAction
  - G4UserRunAction
  - G4UserEventAction
  - G4UserStackingAction
  - G4UserTrackingAction
  - G4UserStepingAction



Detector Simulation

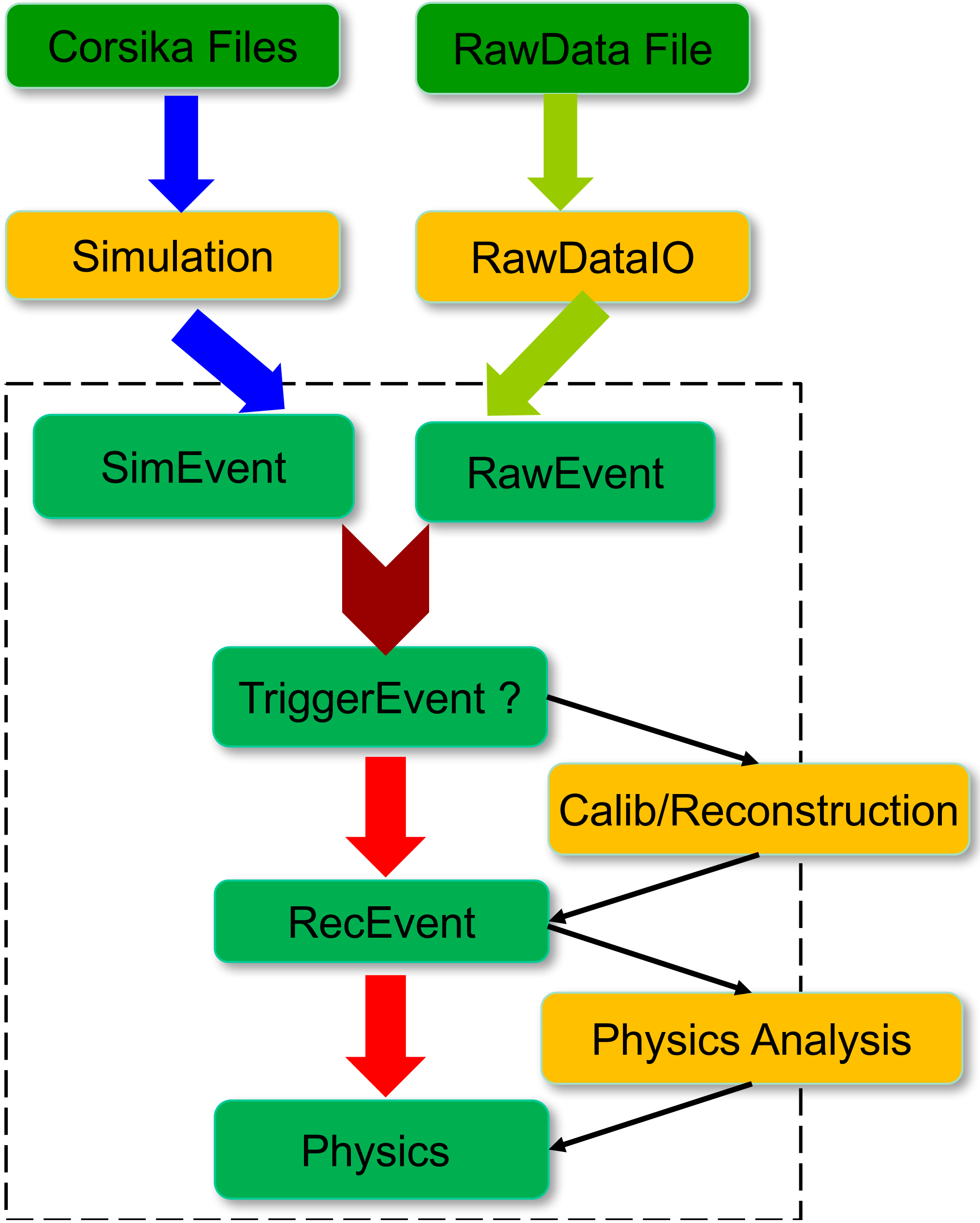
- KM2A fast simulation （Ye Liu, Teng Li)
  - Two algorithms: KM2ADetSimAlg and KM2ARecAlg
  - http://svn.lhaaso.ihep.ac.cn/repos/offline/trunk/FastSimulation/

- WFCTA Simulation (LingLing Ma, Teng Li)
  - One algorithm: WFCTADetSimAlg
  - http://svn.lhaaso.ihep.ac.cn/repos/offline/trunk/Simulation/WFCTASim/

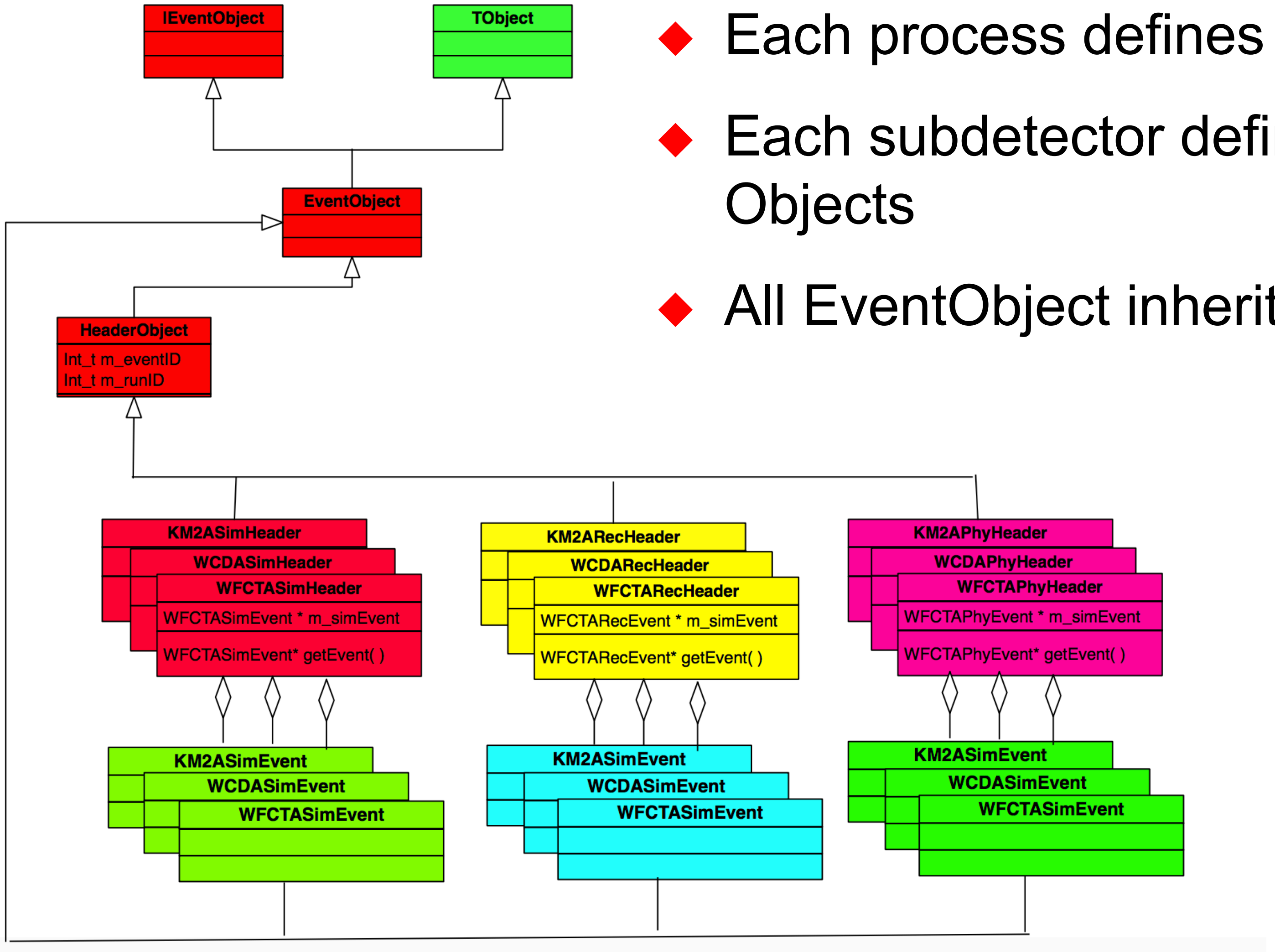
Detector Simulation

- Upgrade of offline software system
  - Geant4.10.04.p01
  - Root:6.10.08
  - Gcc:4.9.4
- KM2A full Simulation (Zhaojing, Songzhan, Wenhao)
  - One DetFactory Service :Km2aSimFactory
  - http://svn.lhaaso.ihep.ac.cn/repos/offline/trunk/Simulation/Km2aSim/
- WCDA Simulation (Hanrong, Min, Zhiguo, Wenhao)
  - One DetFactory Service: WcdaSimFactory for simulation without PMT
  - http://svn.lhaaso.ihep.ac.cn/repos/offline/trunk/Simulation/WcdaSim/
  - One DetFactory Service: Wcda2SimFactory for PMT Simulation
  - http://svn.lhaaso.ihep.ac.cn/repos/offline/trunk/Simulation/Wcda2Sim/

Event Data Model



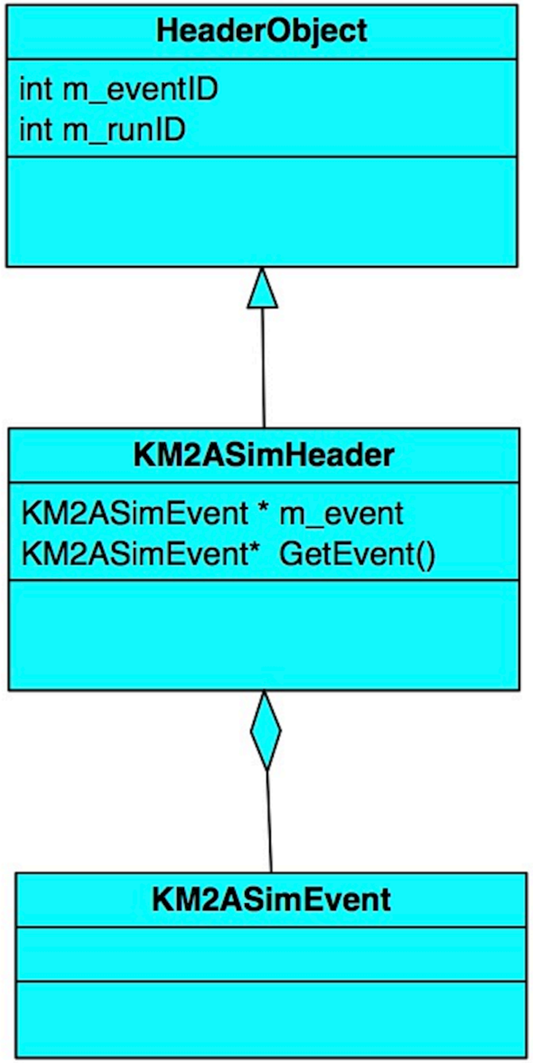
Current Design of Event Data Model



- Each process defines their Event Objects
- Each subdetector defines their Event Objects
- All EventObject inherits from TObject.

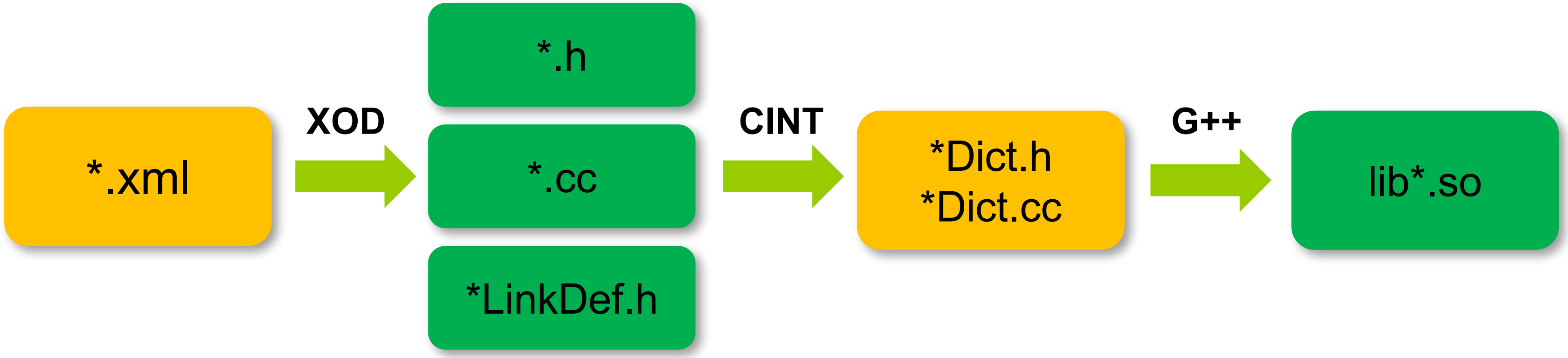
Example: KM2A Simulation Event Object

- Each event data consists of two parts:
  - HeaderObject
  - EventObject
  - Separate meta data from event data
- HeaderObject
  - defines specific requirements for LHAASO
  - Some “tag” information to speed up event selection and build corrections between different objects
- KM2ASimHeader has a pointer to the KM2ASimEvent
- KM2ASimEvent holds event information

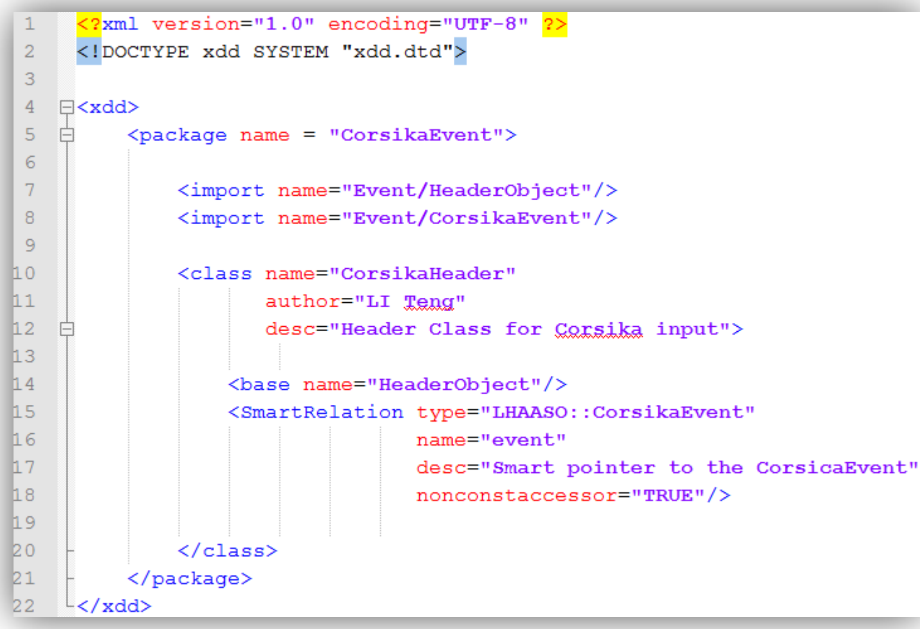


Event Data Definition with XML file

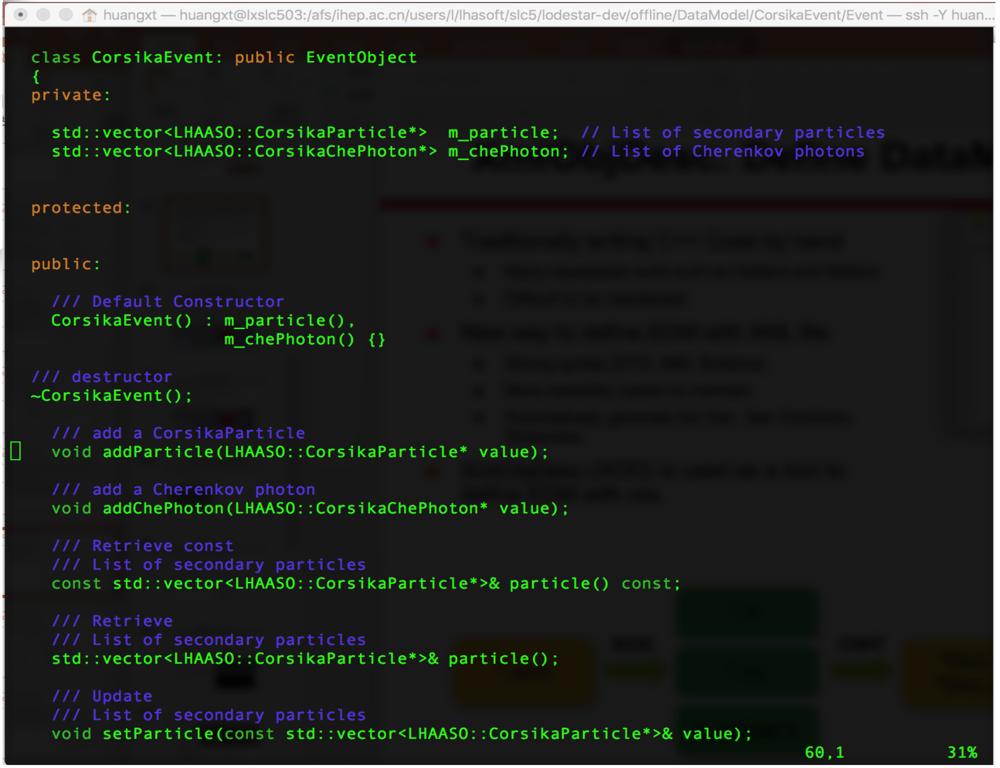
- Traditionally writing C++ Code by hand
  - Many repeatable work such as Getters and Setters
  - Difficult to be maintained
- A new method to define EDM with XML file
  - Strong syntax (DTD, XML Schema)
  - More readable, easier to maintain
  - Automatically generate the Get-, Set- functions, Streamers
- XmlObjDesc (XOD) is used as a tool to define EDM with XML



XOD Example: CorsikaEvent



```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xdd SYSTEM "xdd.dtd">

<xdd>
    <package name = "CorsikaEvent">

        <import name="Event/HeaderObject"/>
        <import name="Event/CorsikaEvent"/>

        <class name="CorsikaHeader"
               author="LI Teng"
               desc="Header Class for Corsika input">

            <base name="HeaderObject"/>
            <SmartRelation type="LHAASO::CorsikaEvent"
                           name="event"
                           desc="Smart pointer to the CorsicaEvent"
                           nonconstaccessor="TRUE"/>

        </class>
    </package>
</xdd>
```



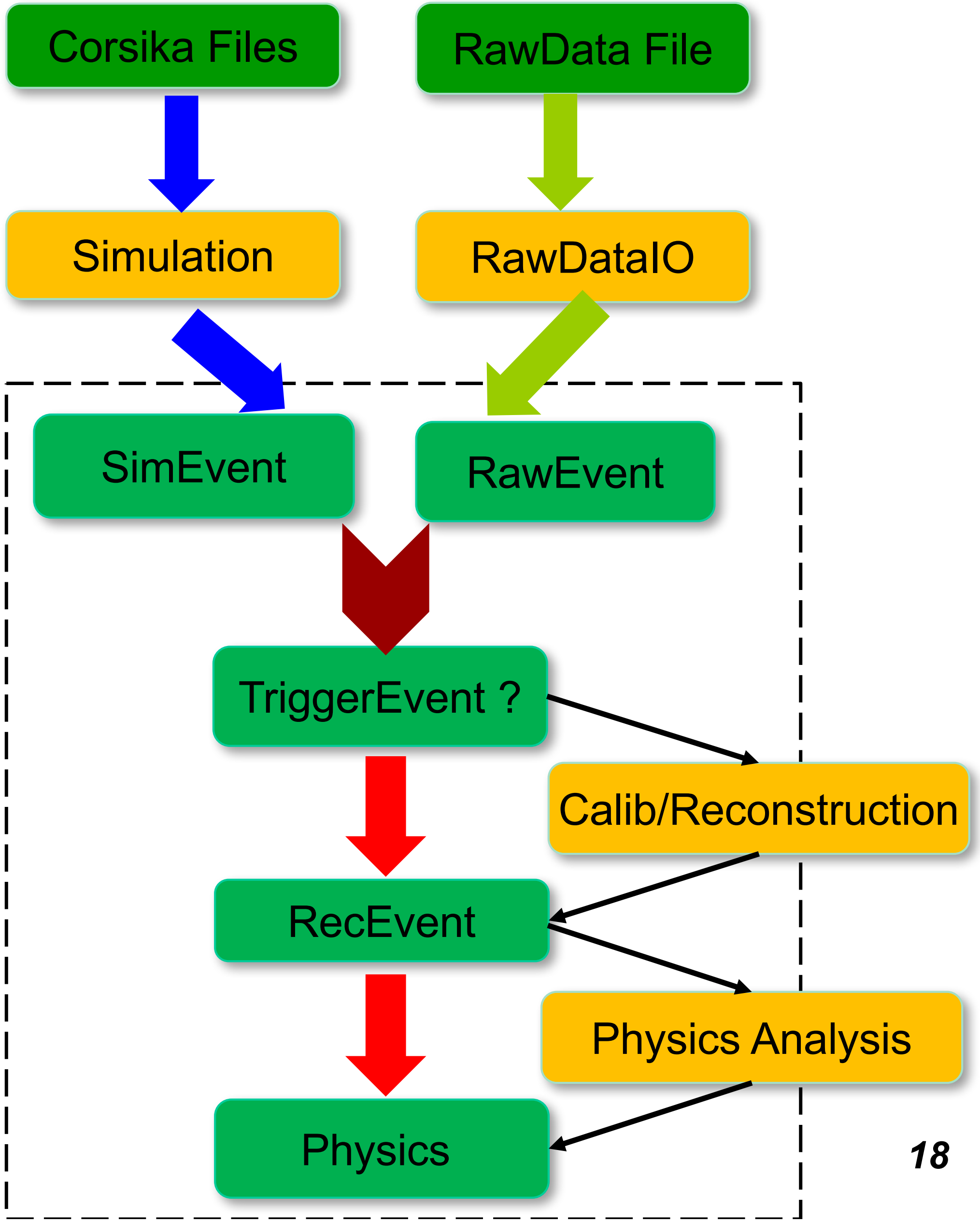
huangxt — huangxt@lxslc503:/afs/ihep.ac.cn/users/l/lhasoft/slc5/lodestar-dev/offline/DataModel/CorsikaEvent/Event — ssh -Y huan...

```
class CorsikaEvent: public EventObject
{
private:

  std::vector<LHAASO::CorsikaParticle*>  m_particle;  // List of secondary particles
  std::vector<LHAASO::CorsikaChePhoton*> m_chePhoton; // List of Cherenkov photons


protected:


public:

  /// Default Constructor
  CorsikaEvent() : m_particle(),
                   m_chePhoton() {}

/// destructor
~CorsikaEvent();

  /// add a CorsikaParticle
  void addParticle(LHAASO::CorsikaParticle* value);

  /// add a Cherenkov photon
  void addChePhoton(LHAASO::CorsikaChePhoton* value);

  /// Retrieve const
  /// List of secondary particles
  const std::vector<LHAASO::CorsikaParticle*>& particle() const;

  /// Retrieve
  /// List of secondary particles
  std::vector<LHAASO::CorsikaParticle*>& particle();

  /// Update
  /// List of secondary particles
  void setParticle(const std::vector<LHAASO::CorsikaParticle*>& value);
                                                                60,1          31%
```

Data Input/Output System

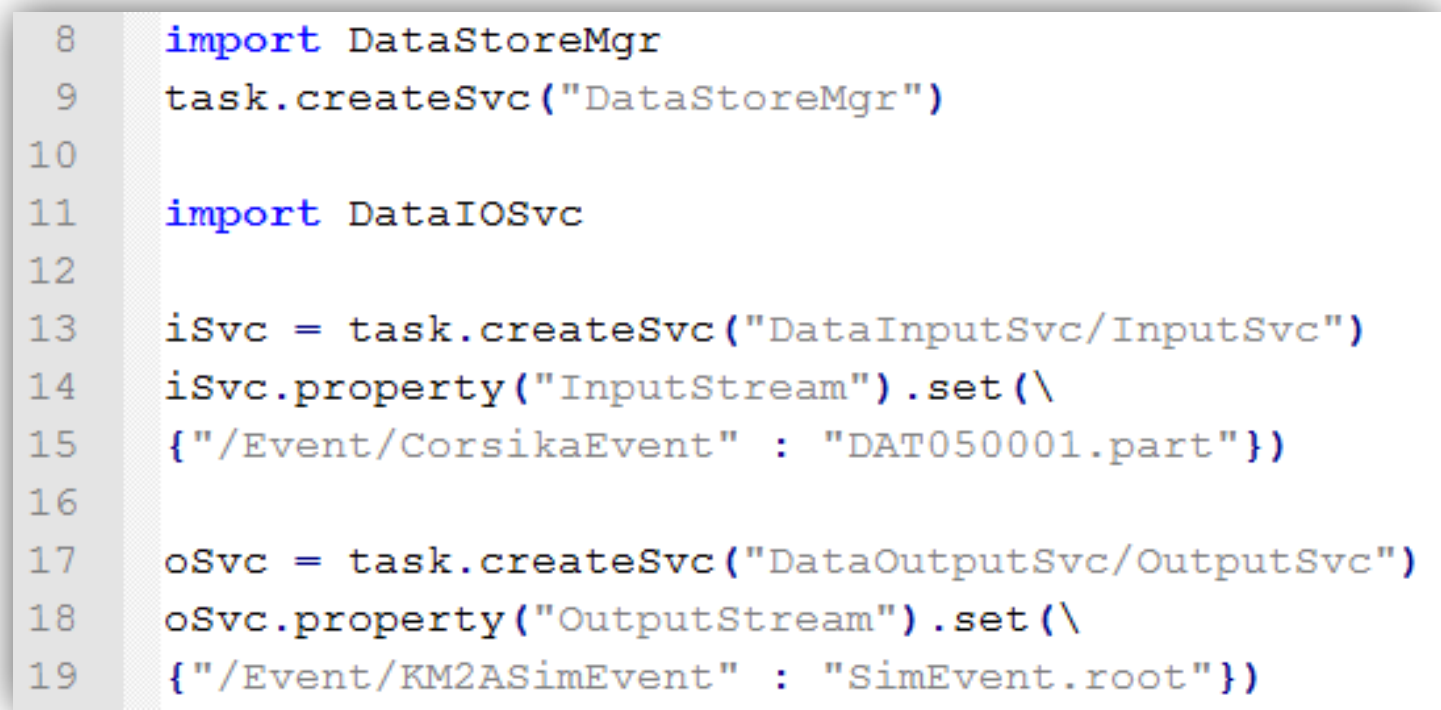
- Two types of IO Systems
- Responsible for reading/ writing event data from/to files.
- Currently support:
  - Corsika Files
  - Root Files
  - Raw Data Files

  (ED Prototype data)



Reading Corsika Files

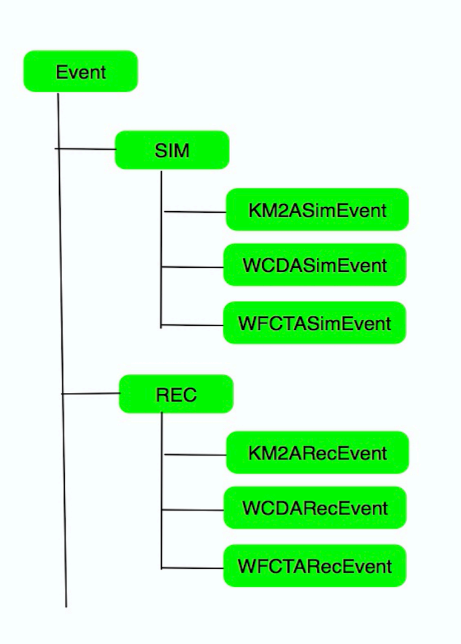
- CorsikaInputSvc
  - Convert the raw corsika file to the DataModel objects
  - Support different types of input files (particles, Cherenkov photons, Longitudinal parameters)
  - Support event-splitting



```
import DataStoreMgr
task.createSvc("DataStoreMgr")

import DataIOSvc

iSvc = task.createSvc("DataInputSvc/InputSvc")
iSvc.property("InputStream").set(\
{"/Event/CorsikaEvent" : "DAT050001.part"})

oSvc = task.createSvc("DataOutputSvc/OutputSvc")
oSvc.property("OutputStream").set(\
{"/Event/KM2ASimEvent" : "SimEvent.root"})
```

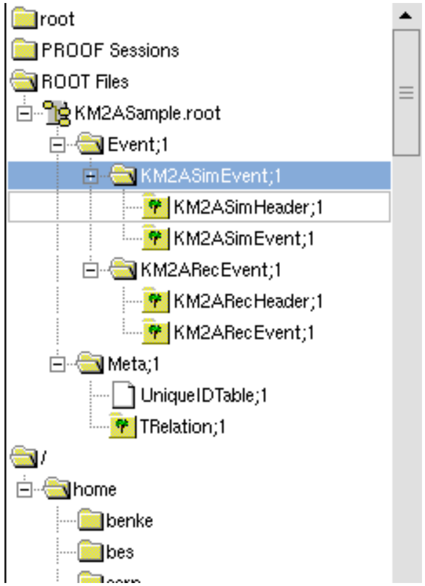
Root Input/Output Service

- RootInputSvc/RootOutputSvc:
  - Read Event Data from Root Files to Data Store
    - Correlation analysis with different sub-detector information
  - Write Event Data from Data Store to Root Files
    - Root Files could be analyzed with root macro scripts



Event
SIM
KM2ASimEvent
WCDASimEvent
WFCTASimEvent
REC
KM2ARecEvent
WCDARecEvent
WFCTARecEvent

Unique Path in Data Store

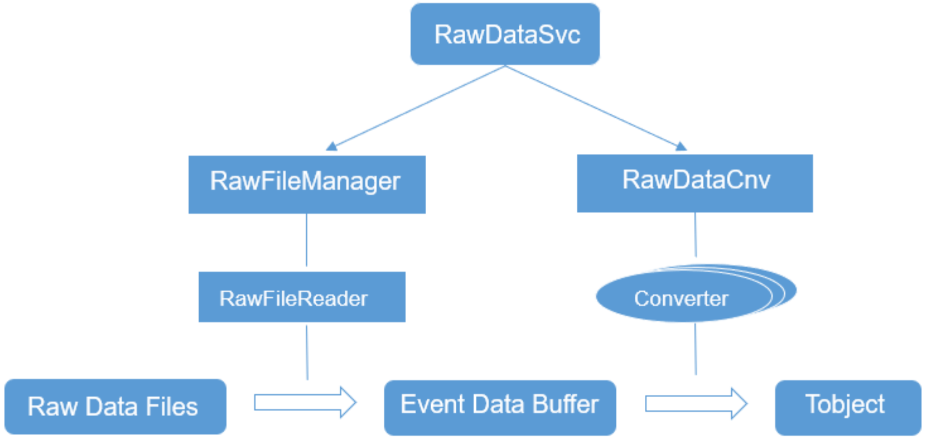


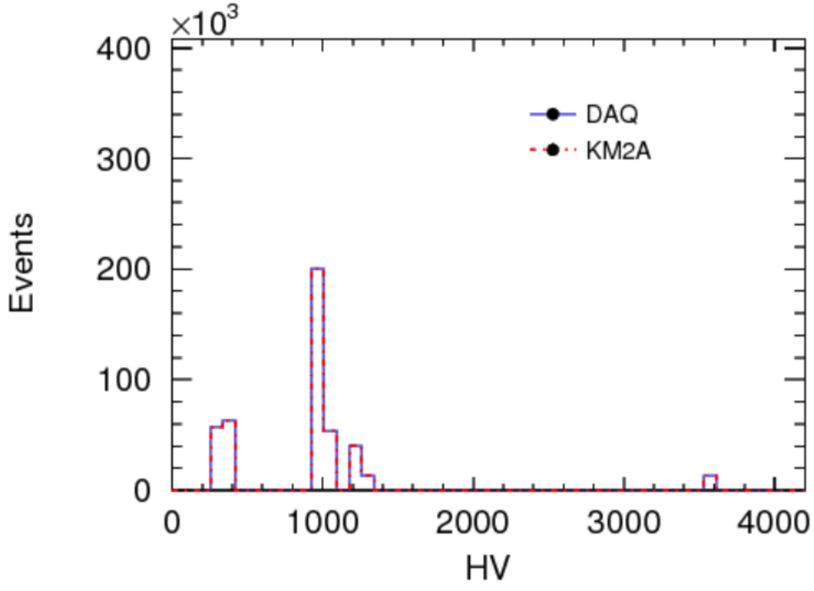
```
root
PROOF Sessions
ROOT Files
  KM2ASample.root
    Event;1
      KM2ASimEvent;1
        KM2ASimHeader;1
        KM2ASimEvent;1
      KM2ARecEvent;1
        KM2ARecHeader;1
        KM2ARecEvent;1
    Meta;1
      UniqueIDTable;1
      TRelation;1
/
  home
    benke
    bes
```

Tree Structure in Root File

Raw Data Input Service

- One new service, RawDataSvc, to manage reading Raw Data from files, decode and write even data into root files

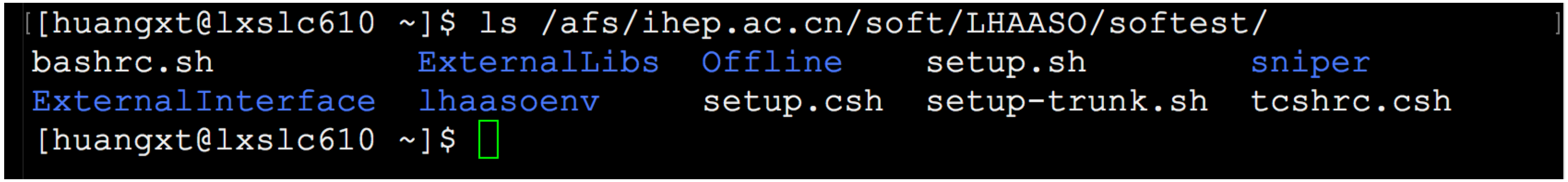




- Svn:http://svn.lhaaso.ihep.ac.cn/repos/offline/trunk/RawIO/

Where to get codes

- A New SVN server has been setup.
  - http://svn.lhaaso.ihep.ac.cn/repos/
    - offline/
    - sniper/
    - cmtlibs/
    - installation/
  - Public account /password (read only): lhaaso / lhaasosvn;123
  - Mail to Wenhao (whyellow@mail.sdu.edu.cn) for new accounts (read or write or both)
- The latest version of LodeStar has been installed at ihep.
  - /afs/ihep.ac.cn/soft/LHAASO/softest/



```
[huangxt@lxslc610 ~]$ ls /afs/ihep.ac.cn/soft/LHAASO/softest/
bashrc.sh          ExternalLibs  Offline    setup.sh        sniper
ExternalInterface  lhaasoenv     setup.csh  setup-trunk.sh  tcshrc.csh
[huangxt@lxslc610 ~]$
```

How to set up LodeStar Environment

- Login ihep computing node
  - ssh –Y username@lxslc6.ihep.ac.cn
- Setup Lodestar environments
  - source /afs/ihep.ac.cn/soft/LHAASO/softest/setup.sh
- Create your own project
  - cmt create_project workarea

    # a new directory, workarea, will be created automatically
  - cd workarea/

    # put all your codes under this directory, workarea

How to Run HelloWorld

- Check out HelloWorld example from svn
  - svn co http://svn.lhaaso.ihep.ac.cn/repos/sniper/trunk/Examples/HelloWorld/

  # a new directory, HelloWorld, will be created with some codes inside

  - cd HelloWorld/cmt

  # a requirements file is used to configure HelloWorld package

  #user need edit it by following this example in user’s package

  - cmt config

  #config package according to requirements file

  - make

  #compile and build HelloWorld package

  - cd ../share

  # a python script to config this job

  - python run.py

How to Run KM2A Detector Simulation

- Check out all simulation pacakge or one detector simulation package
  - svn co http://svn.lhaaso.ihep.ac.cn/repos/offline/trunk/Simulation/
  - Svn co http://svn.lhaaso.ihep.ac.cn/repos/offline/trunk/Simulation/Km2aSim
- Configure, compile and build local package
  - Cd Simulation/Km2aSim/cmt
  - cmt config
  - Make
- Setup environments and run it
  - source setup.sh
  - cd ../share
  - python run.py

Summary

- Offline software system have been upgraded
  - Including Geant4, ROOT and GCC
- KM2A, WCDA and WFCTA have been integrated with LodeStar
  - More testing and optimization are needed
- All simulation codes have been committed into SVN
  - Suggesting all updates should start from the codes of SVN
- Preliminary RawDataIO has been implemented
  - Keep the part related with data format as flexible as possible

Planning

- Begin integration of reconstruction into LodeStar
- Setup the whole chain from Corsica (Raw) data to physics analysis
- Prepare more examples for tutorial
- Release next official version for the collaboration in Dec. 2018

Thanks for your attention !