Progress of LodeStar and It's Application

Wenhao Huang, Xingtao Huang, Qiyun Li, Na Yin

Shandong University

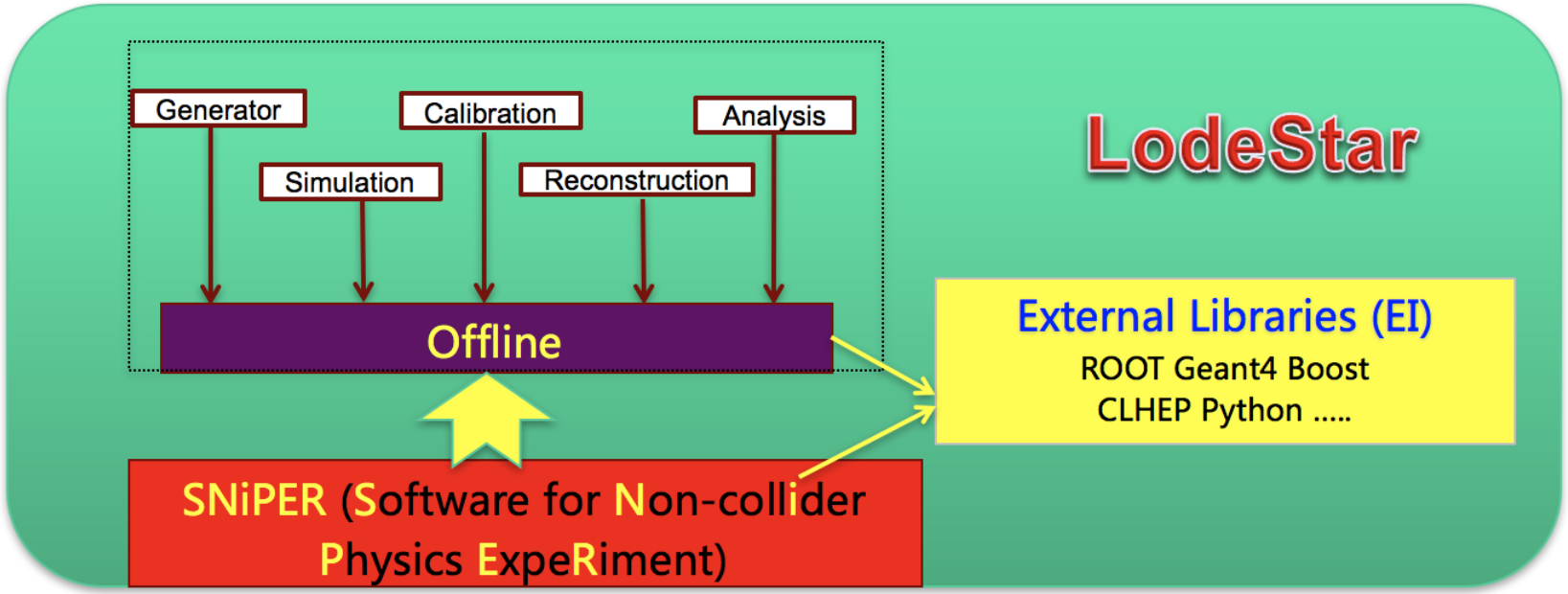
2019.10.23-25 Zhuhai

Outline

- Overview of LodeStar
- Progresses since Last Collaboration Meeting
- Current Status
- Summary & Plan

Whole Design of LodeStar

- LodeStar
  - **L**HAASO **O**ffline **D**ata Proc**e**ssing **S**oft**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



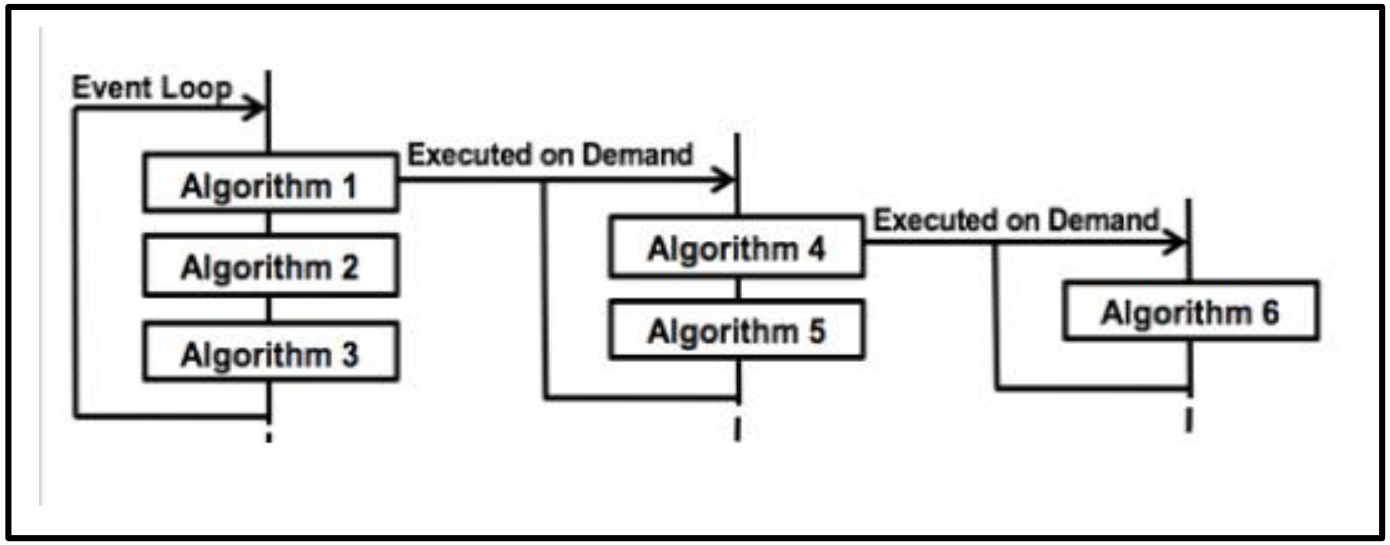
Features of LodeStar

- Modulization/Standardization
  - Algorithms, services and tools, which are most frequently used by users
    - Independent Function
    - Unified Interface

- Main functionality
  - **Unified event data definition**
  - **Unified Event management**
  - **Unified User interface and common Services**
    - **Python script for job configuration**
    - **Services for booking histograms or ntuples, database acccessing,…**
  - **Unified Event loop**
    - user only concentrate in the processing of ONE event.

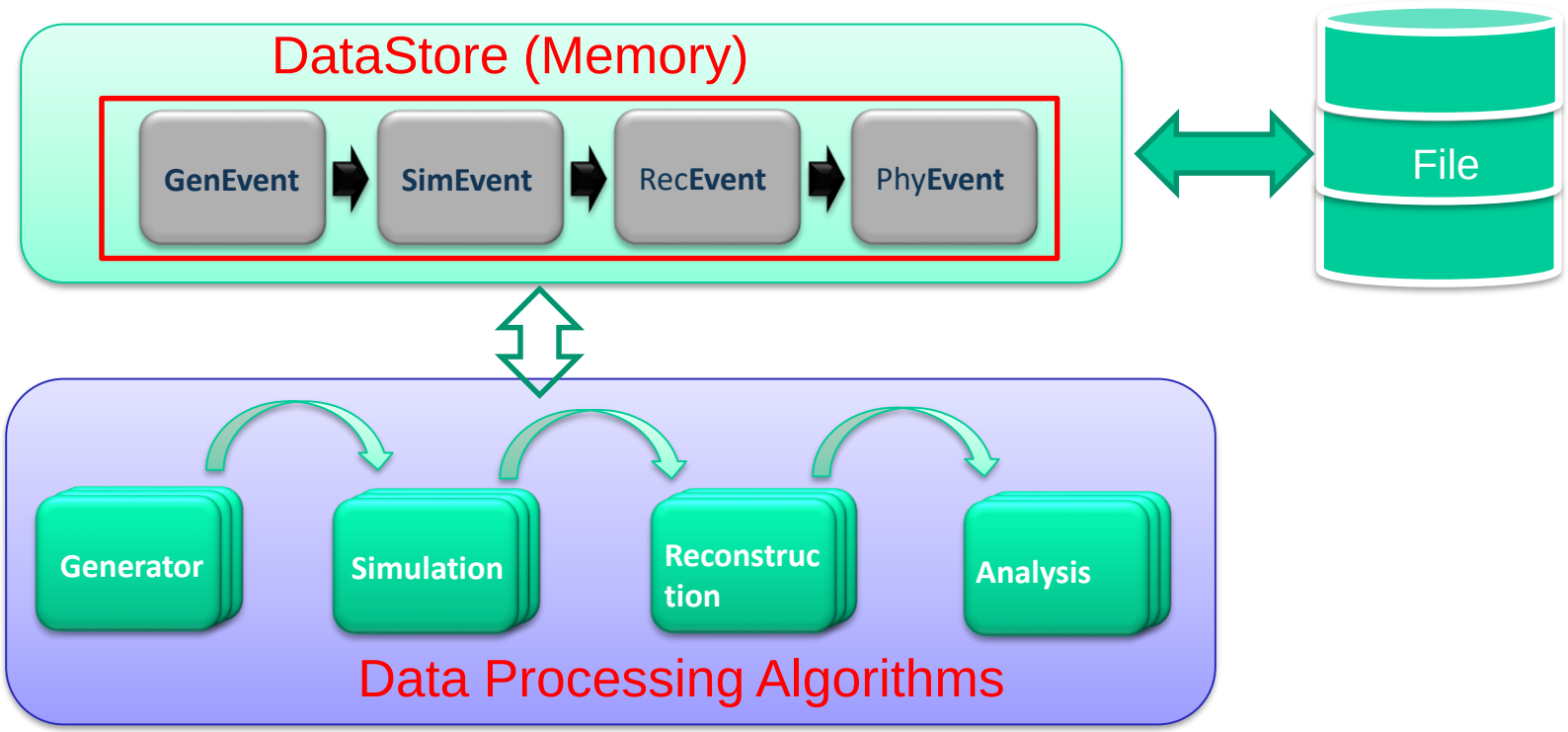
Event Loop

- Event loop is implemented by **algorithms**.
  - An unit of codes for Data Processing
    - the calculation during event loop
    - Most frequently used by users
- Algorithms are easy to combine
  - Managed by framework
  - Unified interface
  - Event sharing

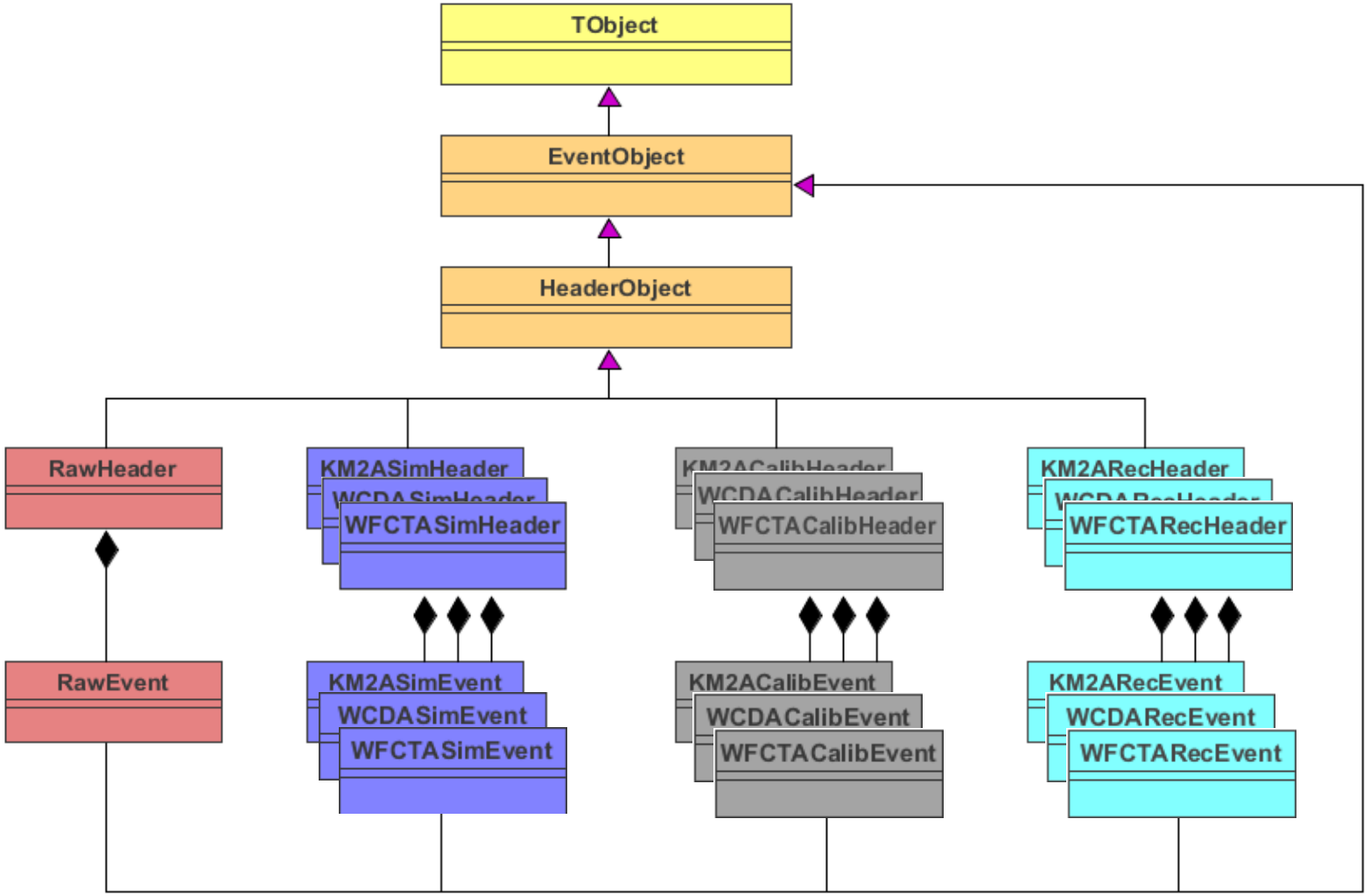


Event Data

- Event data:
  - Processed by algorithms
  - Defined by Event data model



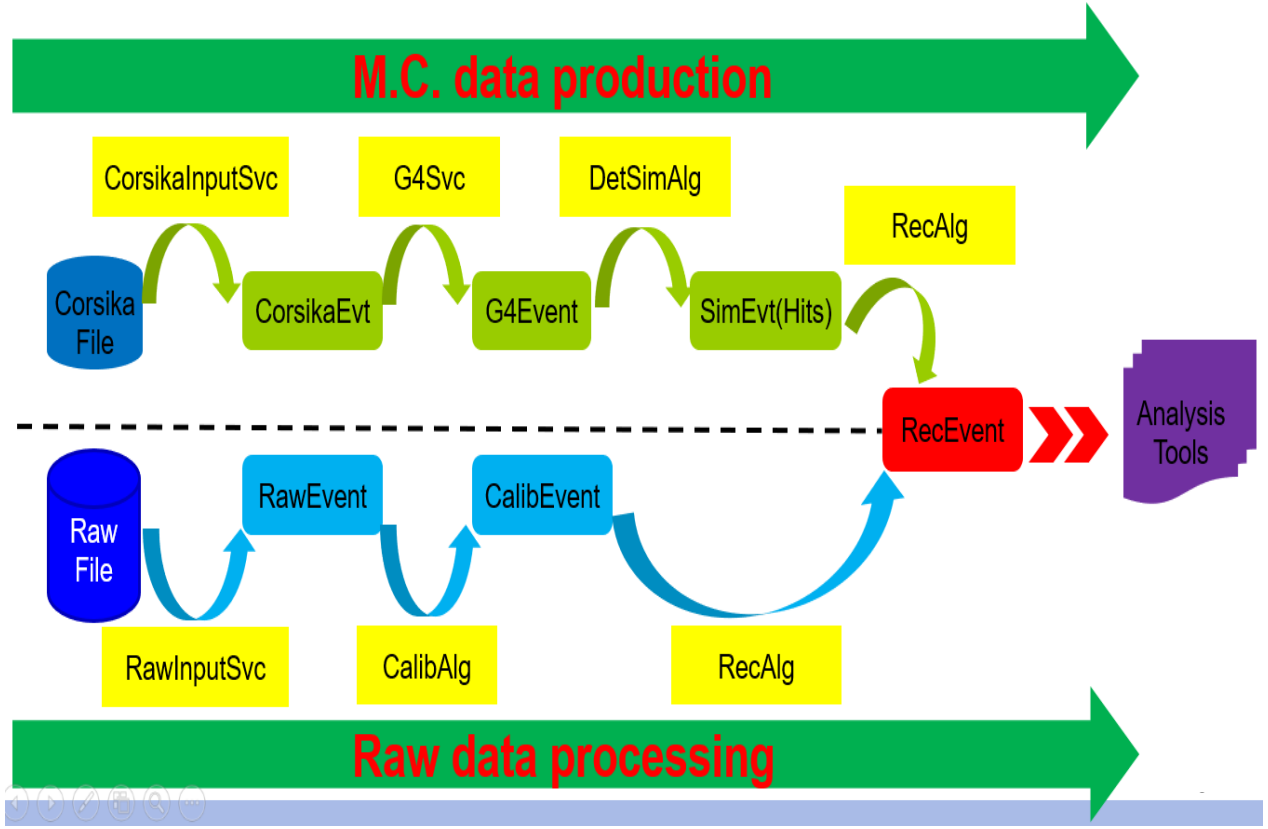
Event Data Model



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

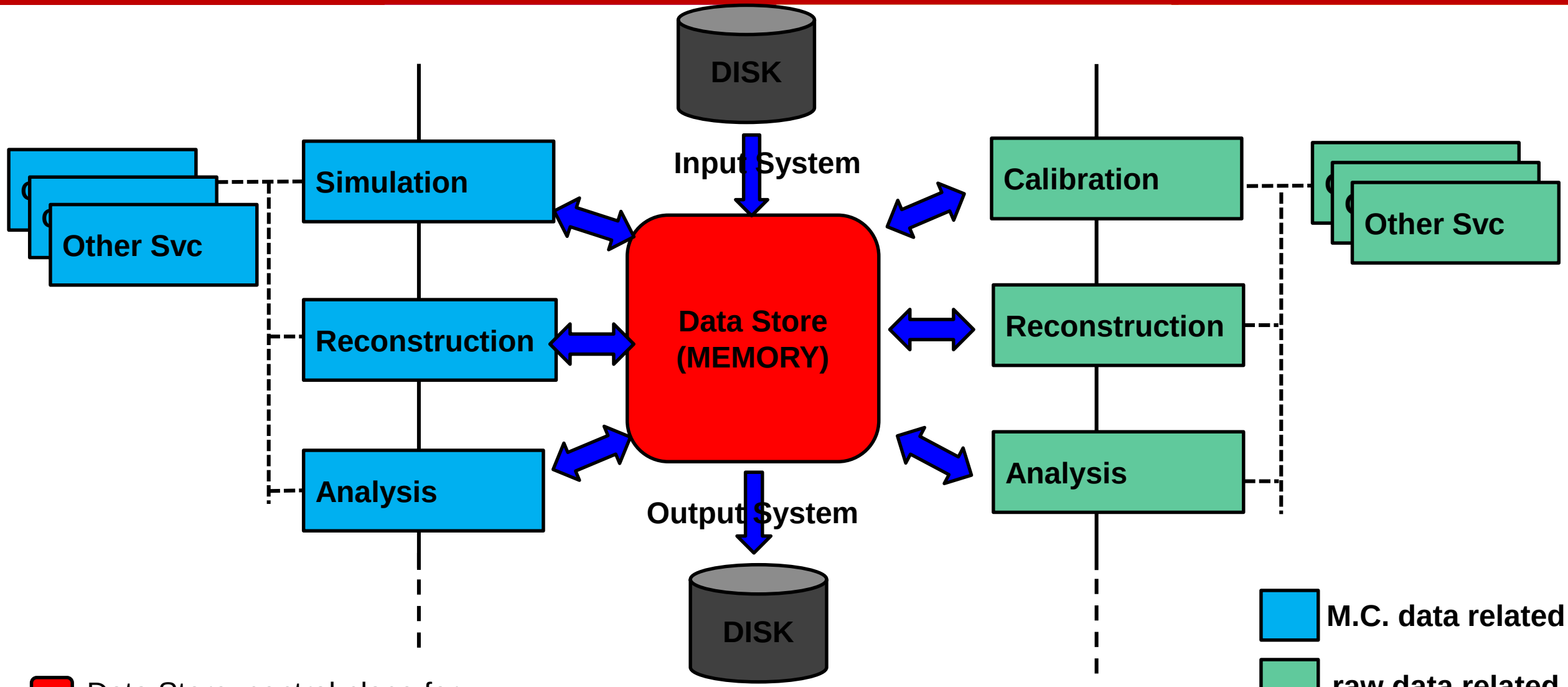
Data Input/Output System

- Three types of Inputs Systems
  - Corsika Input system (Corsika Data)
  - Raw Input system (Raw Data)
  - Root Input System(ROOT Data)



- One Unified Root Output system
  - Writing all Event data from Memory into Root Files

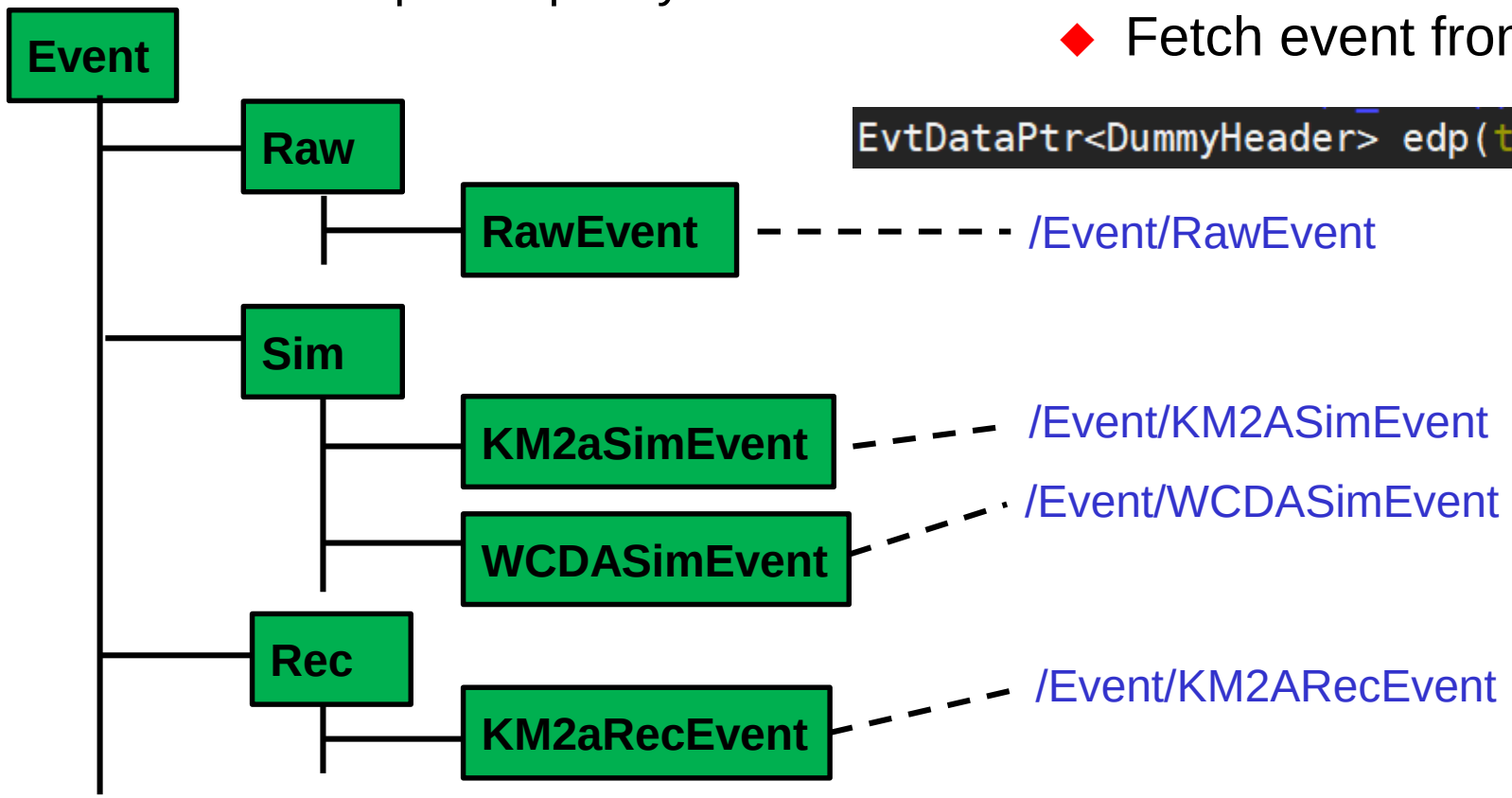
Event Management and Sharing in Data Store



Data Store: central place for holding and sharing Event data

Management of Event data in DataStore

- DatastoreMgr and “key”
  - Manage the logical path of Event data
  - Manage the lifecycle of Event data
  - Coordinate input/output system



- Put event into data store

```
SniperPtr<IDataStoreMgr> mMgr(getParent(), "DataStoreMgr");
mMgr->adopt(Header, "/Event");
```

- Fetch event from data store

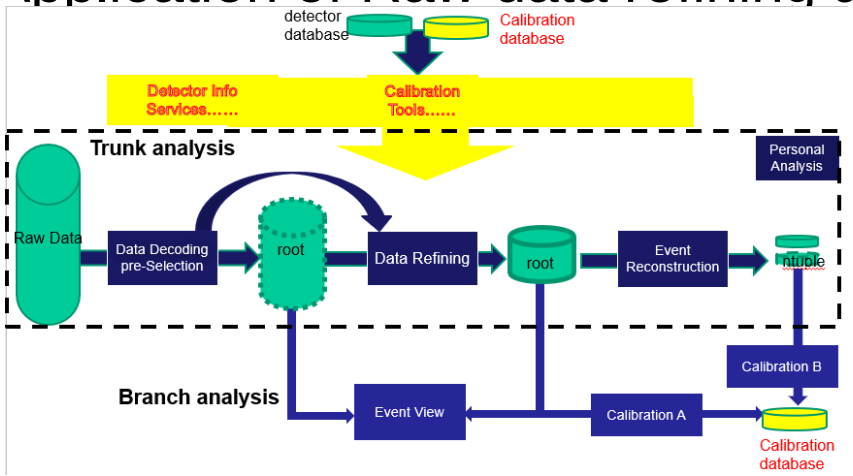
```
EvtDataPtr<DummyHeader> edp(this->getRoot(),"/Event/DummyEvent");
```

Progress since last collaboration meeting

- KM2A simulation(Wenhao Huang, Jing Zhao)
  - Corsika input interface
    - Corsika event may split into smaller one contains certain numbers of sub-particles
  - System memory check
    - Optimized algorithm
- Input/Output system interface
  - Event-steps
  - Event-starts
- WFCTA simulation(Lingling Ma, Wenhao Huang)
  - Data Model updating
  - Data flows from simulation to analysis

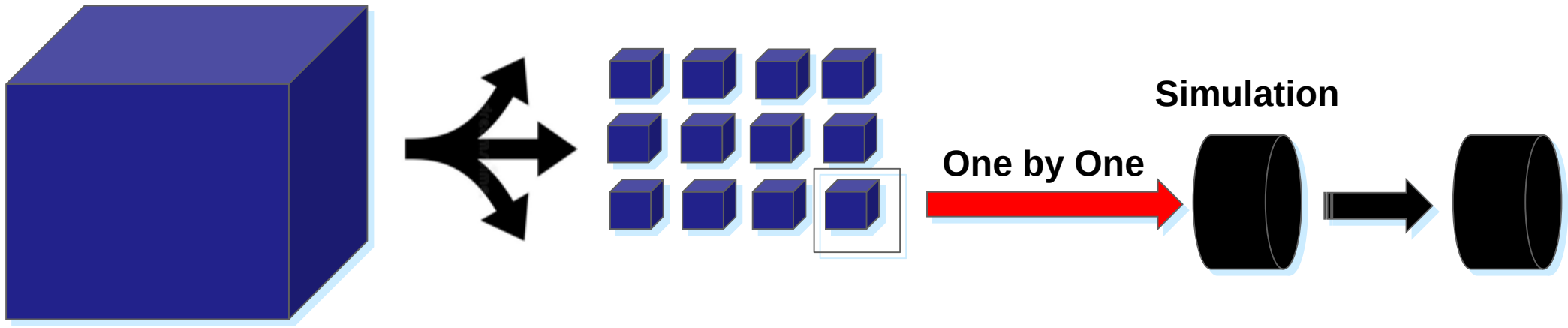
Progress since last collaboration meeting

- KM2A time calibration
  - Integration into LodeStar (Qiyun Li, Yuncheng Nan)
    - Product calibration data in framework way
- Management of calibration data
  - Stored in database
  - Calibration data service
- Application of Raw data refining and reconstruction (followed talk by Chenguang, Zhu)



Optimizations in KM2A Simulation

- Corsika Input



complete corsika event

split corsika event

- Split corsika event is the unit of corsika data proceeding

KM2A Time Calibration

- Produce calibration data
  - Average algorithm: KM2ATimAvg
  - Calibration algorithm: KM2ATimCal
- Calibration database
  - http://lhaasodb.hepg.sdu.edu.cn:8888/phpmyadmin/
  - Structure and information are under discussing.

+ 选项

	du_id	id	time	Time_Start	Time_End
编辑 复制 删除	1	2	10.31543794	2019-03-05 00:00:00	2019-03-05 00:00:00
编辑 复制 删除	2	3	9.356503145	2019-03-05 00:00:00	2019-03-05 00:00:00
编辑 复制 删除	3	4	17.54704151	2019-03-05 00:00:00	2019-03-05 00:00:00
编辑 复制 删除	4	9	17.73587896	2019-03-05 00:00:00	2019-03-05 00:00:00
编辑 复制 删除	5	10	15.14012148	2019-03-05 00:00:00	2019-03-05 00:00:00

Calibration Data Service

- DatabaseCalibSvc(calibration database service)
  - Fetch information from database according to certain “key” information
  - For KM2A Time is “timestamp”

```
SniperPtr<DatabaseCalibSvc> hs(getParent(), "hSvc");
    if (hs.valid()) {
        double calibtime = hs->GetKM2ACalibTime(200, "2019-06-20");
    }
```

“key” information: ”timestamp”

- DetcPosSvc(detector position service, by Yonggang, Wang)
  - Get detector position using detector id

```
SniperPtr<DetcPosSvc> hs(getParent(), "hSvc");
    if (hs.valid()) {
     double x =  hs->GetDetXByID(32);
     double y =  hs->GetDetXByID(32);
     double z =  hs->GetDetXByID(32);
    }
```

“key” information: ”detector id”

Current status

- Detector simulation works well in framework
  - KM2A fast simulation
  - KM2A full simulation
  - WCDA simulation
  - WFCTA simulation
- Simulation data Reconstruction
  - KM2A core position and direction reconstruction
  - WCDA core position reconstruction
- Calibration
  - KM2A time calibration
- Raw data refining and construction

Tutorial: Examples

- 14 Examples in backup corresponding the most common functions
- Tutorials
  - Example1(HelloWorld)
    - Basic operations to run an algorithm
  - Example2(Root Writer)
    - An easy way to use root-related functions: define histogram or Ntuple…
  - Example3(Data Model)
    - Learn about generate data model with xml files

Tutorial: Examples in backup

- Tutorials
  - Example4(Data I/O system)
    - Learn about event transmission between memory and disk
  - Example5~Example6(Specified Input System)
    - How to read corsika files
    - How input system to read raw files
  - Example8~Example12(Detector Simulation: KM2A, WCDA, WFCTA…)
    - How to run detector simulation
  - Example13~Example14(Reconstruction)
    - How to run reconstruction

Detector Simulation

- KM2A fast simulation （Ye Liu, Teng Li)
  - Fast simulatin algorithm: KM2ADetSimAlg
  - Reconstruction algorithm: KM2ARecAlg
  - http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/FastSimulation
  - Included in Example 8
- KM2A full Simulation (Jing Zhao, Songzhan Chen, Wenhao Huang)
  - One DetFactory Service :KM2ASimV2Factory
  - http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/KM2ASimV2
  - Included in Example 9

Detector Simulation

- WCDA Simulation (Hanrong Wu, Min Zha, Zhiguo Yao, Wenhao Huang)
  - One DetFactory Service: WcdaSimFactory for simulation without PMT
  - http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/WcdaSim/
  - One DetFactory Service: Wcda2SimFactory for PMT Simulation
  - http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/Wcda2Sim/
  - Included in Example 10 and Example 11
- WFCTA Simulation (LingLing Ma, Wenhao Huang, Teng Li)
  - One algorithm: WFCTADetSimAlg
  - http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/WFCTASim/
  - Included in Example 12

Calibration

- KM2A time calibration
  - Average algorithm: KM2ATimAvg
  - Calibration algorithm: KM2ATimCal
  - http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/calibration/KM2ATimCalAlg
  - Included in Example 7

Reconstruction

- KM2A core position and direction reconstruction(Songzhan Chen, Wenhao Huang)
  - One algorithm: Km2aRec
  - http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Reconstruction/KM2Arec
  - Included in Example 14

- Wcda core position reconstruction(Xiaojie Wang, Wenhao Huang)
  - One algorithm: Km2aRec
  - http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Reconstruction/WCDArec
  - Included in Example 13

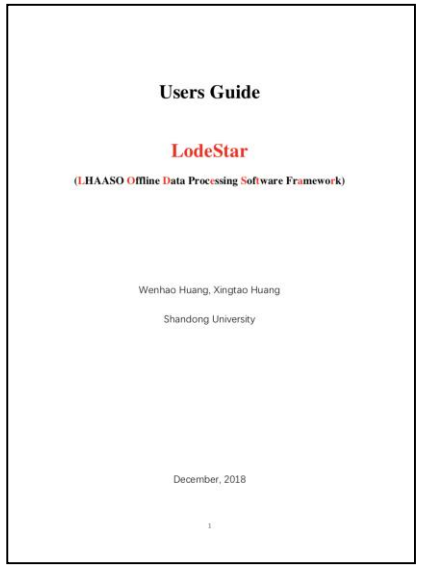
Latest version and documentation

- All codes have been committed to svn.
  - http://svn.lhaaso.ihep.ac.cn/LodeStar
- The latest version of LodeStar has been installed at ihep.
  - /afs/ihep.ac.cn/soft/LHAASO/LodeStar-SLC6/Pre-Release/L19-Pre1_v1r1

```
-bash-4.1$ cd /afs/ihep.ac.cn/soft/LHAASO/LodeStar-SLC6/Pre-Release/L19-Pre1_v1r1/
-bash-4.1$ ls
bashrc.sh          ExternalLibs  offline         setgcc494.sh  setup.sh        sniper
ExternalInterface  lhaasoenv     setgcc494.csh   setup.csh     setup-trunk.sh  tcshrc.csh
```

- **Doc-Db**

  581-v3


Users Guide

LodeStar

(LHAASO Offline Data Processing Software Framework)

Wenhao Huang, Xingtao Huang

Shandong University

December, 2018

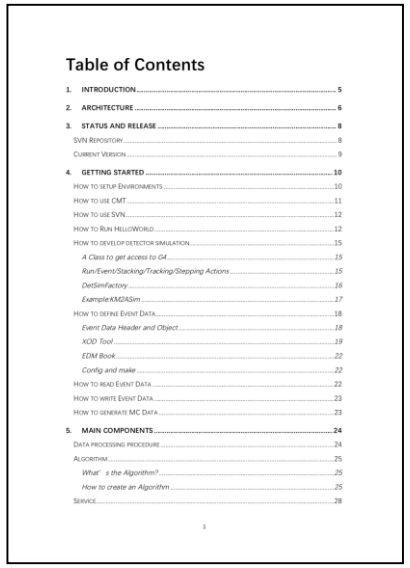
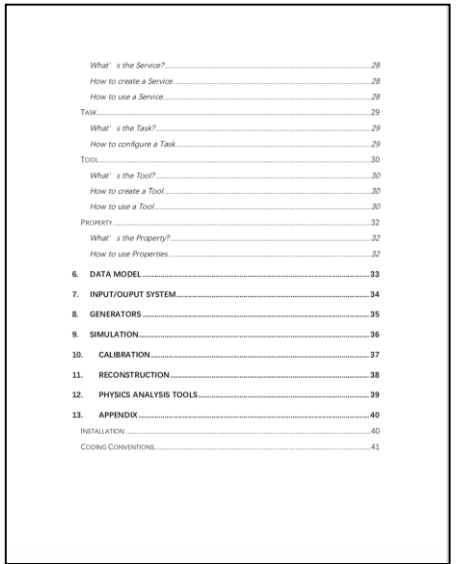


Table of Contents

1. INTRODUCTION 5
2. ARCHITECTURE 6
3. STATUS AND RELEASE 8
   SVN REPOSITORY 8
   CURRENT VERSION 9
4. GETTING STARTED 10
   HOW TO SETUP ENVIRONMENTS 10
   HOW TO USE CMT 11
   HOW TO USE SVN 12
   HOW TO RUN HELLOWORLD 12
   HOW TO DEVELOP DETECTOR SIMULATION 15
   A Class to get access to G4 15
   Run/Event/Stacking/Tracking/Stepping Actions 15
   DetSimFactory 16
   ExampleXM2ASim 17
   HOW TO DEFINE EVENT DATA 18
   Event Data Header and Object 18
   XOD Tool 19
   EDM Book 22
   Config and make 22
   HOW TO READ EVENT DATA 22
   HOW TO WRITE EVENT DATA 23
   HOW TO GENERATE MC DATA 23
5. MAIN COMPONENTS 24
   DATA PROCESSING PROCEDURE 24
   ALGORITHM 25
   What' s the Algorithm? 25
   How to create an Algorithm 25
   SERVICE 28

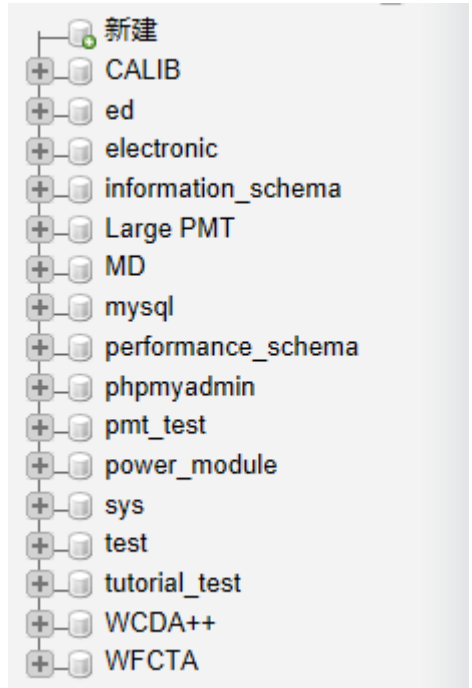

What' s the Service? 28
How to create a Service 28
How to use a Service 28
TASK 29
What' s the Task? 29
How to configure a Task 29
TOOL 30
What' s the Tool? 30
How to create a Tool 30
How to use a Tool 30
PROPERTY 32
What' s the Property? 32
How to use Properties 32
6. DATA MODEL 33
7. INPUT/OUPUT SYSTEM 34
8. GENERATORS 35
9. SIMULATION 36
10. CALIBRATION 37
11. RECONSTRUCTION 38
12. PHYSICS ANALYSIS TOOLS 39
13. APPENDIX 40
    INSTALLATION 40
    CODING CONVENTIONS 41

Database and elog related

- Database
  - http://lhaasodb.hepg.sdu.edu.cn:8888/phpmyadmin/
  - People in charge: Jia Liu(jialiu@ihep.ac.cn) Wenhao Huang(whyellowbred@foxmail.com)
- Elog
  - http://svn.lhaaso.ihep.ac.cn:8080/
  - O&M: Wenhao Huang(whyellowbred@foxmail.com)

Several logbooks are defined on this host.
Please select the one to connect to:

Logbook	Entries	Last submission
demo General Linux Tips & Tricks	4	Sat Aug 24 14:02:41 2019 by huangxt
test General Linux Tips & Tricks	2	Fri Oct 26 15:35:38 2018 by huangwh
edlog General Linux Tips & Tricks	12	Fri Mar 15 14:29:18 2019 by Liujia
mdlog Muon Detector group'elog	143	Mon Oct 21 15:17:48 2019 by 罗智，高启
wcdalog WCDA's logbook.	441	Tue Oct 8 14:17:03 2019 by liuc
wfctalog General Linux Tips & Tricks	19	Thu Jun 13 11:20:23 2019 by youzhiyong
gflog General Linux Tips & Tricks	0	-
daqlog General Linux Tips & Tricks	21	Fri May 10 19:36:05 2019 by Gu Minhao
omlog General Linux Tips & Tricks	0	-



Summary & Plan

- LodeStar provides the key functions for LHAASO Data processing
- M.C Data flow has been successfully integrated into Lodestar
  - Detector simulation (KM2A, WCDA and WFCTA)
  - Calibration(KM2A)
  - Reconstruction (KM2A, WCDA, WFCTA)
- Raw Data Flows are also integrated into LodeStar.
  - Raw data input
  - Calibration
  - Refining and construction
- Welcome more collaborators to use LodeStar
  - Running examples in the backup is a good way to start

- Wenhao Huang
  - whyellowbred@foxmail.com
- Xingtao Huang
  - huangxt@sdu.edu.cn

Do not hesitate to contact us!

Thank you

Backup

How to set up LodeStar Environment

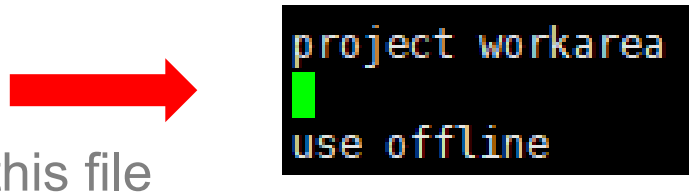
- Login ihep computing node
  - ssh –Y your_user_name@lxslc6.ihep.ac.cn
- Setup Lodestar environments
  - source /afs/ihep.ac.cn/soft/LHAASO/LodeStar-SLC6/Pre-Release/L19-Pre1_v1r1/setup.sh

  #your can also copy this script into your own directory
- 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
  - cd cmt/
  - vim project.cmt

  #add “use offline” in this file



Example 1: Hello World

- Very simple algorithm

- We will learn
  - How to create and compile a LodeStar package
  - How to config and run an algorithm via python script
  - A little about cmt requirement
  - The way of 3 abstract functions called by framework
  - How to declare properties and do python binding

How to Run HelloWorld

- cd path_to_workarea

#Return to directory “workarea”. You can also do: cd ${CMTPATH%%:*}

- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/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

- source setup.sh

#make the library usable

- cd ../share

a python script to confige this job

- python run.py

The HelloWorld Algorithm

- An algorithm should be derived from AlgBase

src/Hello.h

```
 4 #include <string>
 5 #include "SniperKernel/AlgBase.h"
 6
 7 class HelloAlg: public AlgBase {
 8
 9     public:
10         HelloAlg(const std::string& name);
11         ~HelloAlg();
12
13         bool initialize();
14         bool execute();
15         bool finalize();
16
17     private:
18         int m_count;
19         std::string m_string;
20
21 };
```

src/Hello.cc

```
 9 HelloAlg::HelloAlg(const std::string& name)
10     : AlgBase(name)
11 {
12     m_count = 0;
13     declProp("VarString", m_string);
14 }
15
16 HelloAlg::~HelloAlg()
17 {
18
19 }
20
21 bool HelloAlg::initialize()
22 {
23     return true;
24 }
25
26 bool HelloAlg::execute()
27 {
28     ++m_count;
29     LogInfo << "executing: " << m_count << std::endl;
30     SniperPtr<HelloSvc> hs(getParent(), "hSvc");
31     if (hs.valid()) {
32         hs->doSomething();
33     }
34     HelloTool* htool = tool<HelloTool>("htool");
35     htool->doSomething();
36     return true;
37 }
38
39 bool HelloAlg::finalize()
40 {
41     return true;
42 }
```

CMT requirement

- There is a directory called cmt in every package.
- And in that directory, there is a file called requirements.

requirements file (HelloWorld/cmt/requirements)

```
1 package HelloWorld
2
3 use SniperKernel v*
4
5 library HelloWorld *.cc
6 apply_pattern install_python_modules
7 apply_pattern linker_library library=HelloWorld
8 apply_pattern install_more_includes more="HelloWorld"
```

- Line 3 means that this package rely on package SniperKernel.

- Now you have no need to know everything about this file, just need to notice line 3…

Properties and python bonding

```cpp
HelloAlg::HelloAlg(const std::string& name)
    : AlgBase(name)
{
    m_count = 0;
    declProp("VarString", m_string);
}
```

without recompiling.

…pass it to algorithm,

- We set new value to “VarString” in python script and…

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

import Sniper

task = Sniper.Task("task")
#task.asTop()
task.setLogLevel(3)

import HelloWorld
alg = task.createAlg("HelloAlg/hAlg")
alg.property("VarString").set("some value")

task.setEvtMax(5)
task.show()
task.run()
```

Example 2: RootWriter

- RootWriter
  - A simple interface to use root-related functions

- We will learn
  - How to define a histogram
  - How to define a Ntuple

How to Run HelloHist

- cd path_to_workarea

#Return to directory “workarea”. You can also do: cd ${CMTPATH%%:*}

- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Examples/HelloHist/

a new directory, HelloHist, will be created with some codes inside

- cd HelloHist/cmt

a requirements file is used to configure HelloHist package

- cmt config

#config package according to requirements file

- make

#compile and build HelloHist package

- source setup.sh

#make the library usable

- cd ../share

a python script to confige this job

- python run.py

The HelloHist Algorithm

src/HelloHist.cc

```cpp
bool HelloHistAlg::initialize()
{
   hist = new TH1D("hist","hist",8,0,8);

   return true;
}

bool HelloHistAlg::execute()
{
    LogInfo << "executing: " << m_count << std::endl;

    ++m_count;
    hist->Fill(m_count);

    return true;
}

bool HelloHistAlg::finalize()
{
    SniperPtr<RootWriter> ws(getParent(), "wSvc");
    if (ws.valid()) {
        ws->attach("Fkey/histDir", hist);
          return true;
    }
    else
         return false;
}
```

Fill 5 times

share/run.py

```python
import Sniper

task = Sniper.Task("task")
#task.asTop()
task.setLogLevel(3)

import HelloHist
alg = task.createAlg("HelloHistAlg/hAlg")

import RootWriter
svc = task.createSvc("RootWriter/wSvc")
svc.property("Output").set({"Fkey" : "HelloHist.root"})

task.setEvtMax(5)
task.show()
task.run()
```

How to Run HelloNtuple

- cd path_to_workarea

#Return to directory “workarea”. You can also do: cd ${CMTPATH%%:*}

- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Examples/HelloNtuple/

a new directory, HelloNtuple, will be created with some codes inside

- cd HelloNtuple/cmt

a requirements file is used to configure HelloNtuple package

- cmt config

#config package according to requirements file

- make

#compile and build HelloNtuple package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python run.py

Example 3: Define your own data model

- XmlObjDesc (XOD)
  - a tool to define EDM with XML

- We will learn
  - How to generate EDM will written xml files
  - Event path in DataStore

How to build your data model

- cd path_to_workarea

#Return to directory “workarea”. You can also do: cd ${CMTPATH%%:*}

- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Examples/DummyEvent/

a new directory, DummyEvent, will be created with some codes inside

- cd DummyEvent/cmt

a requirements file is used to configure DummyEvent 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 DummyEvent package

- source setup.sh

#make the library usable

Your own data model

- After compilation, there show be like this

```
-bash-4.1$ cd DummyEvent/
-bash-4.1$ ls
amd64_linux26  cmt  Event  src  xml
```

- In src directory, there would be a file named “DummyEDMDef.cc”.

```
-bash-4.1$ cd src/
-bash-4.1$ ls
DummyEDMDef.cc      DummyEventDict_rdict.pcm  DummyHeaderDict.cc
DummyEvent.cc       DummyEventLinkDef.h       DummyHeaderDict_rdict.pcm
DummyEventDict.cc   DummyHeader.cc            DummyHeaderLinkDef.h
```

- Notice that the path is defined here and we will use it in Example 4.

```
#include "EDMUtil/BookEDM.h"
LHAASO_BOOK_EDM(LHAASO::DummyHeader, LHAASO::DummyEvent, 999, /Event/DummyEvent);
```

path in DataStore

Example 4: Data model IO system(RootIO)

- This Example is based on Example3

- We will define algorithms:
  - To learn about data I/O system
    - create DummyEvent and write them into disk at root files.
    - read the root file above and fetch the results.

How to use your data model(1)

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Examples/IOTestAlg/

a new directory, IOTestAlg, will be created with some codes inside

- cd IOTestAlg/cmt

a requirements file is used to configure IOTestAlg package

- cmt config

#config package according to requirements file

- make

#compile and build IOTestAlg package

- source setup.sh

#make the library usable

How to use your data model(2)

- cd ../share

#Python scripts in this directory

- Python EdmWrite.py

#write out root file via output system

- Python EdmRead.py

#read the root file above via input system

```
iSvc = AlgTask.createSvc("RootInputSvc/InputSvc")
iSvc.property("InputStream").set({"/Event/DummyEvent" : "DummyEvent.root"})
```

```
oSvc = AlgTask.createSvc("RootOutputSvc/OutputSvc")
oSvc.property("OutputStream").set({"/Event/DummyEvent" : "DummyEvent.root"})
```

```
EvtDataPtr<DummyHeader> edp(this->getRoot(),"/Event/DummyEvent");
```

Example 5: Data model IO system(CorsikaInput)

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Examples/CorsikaIOTestAlg/

a new directory, CorsikaIOTestAlg, will be created with some codes inside

- cd CorsikaIOTestAlg/cmt

a requirements file is used to configure CorsikaIOTestAlg package

- cmt config

#config package according to requirements file

- make

#compile and build CorsikaIOTestAlg package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python run.py

Example 6: Data model IO system(raw data Input)

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Examples/RawIOTestAlg/

a new directory, RawIOTestAlg, will be created with some codes inside

- cd RawIOTestAlg/cmt

a requirements file is used to configure RawIOTestAlg package

- cmt config

#config package according to requirements file

- make

#compile and build RawIOTestAlg package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python run.py

Example 7: KM2A time calibration

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Calibration/KM2ATimCal/

a new directory, KM2ATimCal, will be created with some codes inside

- cd KM2ATimCal/cmt

a requirements file is used to configure KM2ATimCal 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 KM2ATimCal package

- source setup.sh

Example 7: KM2A time calibration

- cd ../share

a python script to config this job

- python average.py -i rootfile.flst -o Timeaverage
- python calib.py -i rootfile.flst –a Timeaverage -o Calibration

  Add “-c Calibration” for iteration, for example :

- python average.py -i rootfile.flst -c calibration -o Timeaverage

Example 8: KM2A fast simulation

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/FastSimulation/KM2ASimulation/

a new directory, KM2ASimulation, will be created with some codes inside

- cd KM2ASimulation/cmt

a requirements file is used to configure KM2ASimulation package

- cmt config

#config package according to requirements file

- make

#compile and build KM2ASimulation package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python run.py

Example 9: KM2A full simulation

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/KM2ASimV2

a new directory, KM2ASimV2, will be created with some codes inside

- cd KM2ASimV2/cmt

a requirements file is used to configure KM2ASimV2 package

- cmt config

#config package according to requirements file

- make

#compile and build KM2ASimV2 package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python DetSim_KM2A.py

Example 10: WCDA simulation(first step)

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/WcdaSim

a new directory, WcdaSim, will be created with some codes inside

- cd WcdaSim/cmt

a requirements file is used to configure WcdaSim package

- cmt config

#config package according to requirements file

- make

#compile and build WcdaSim package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python DetSim_WCDA_2.py [input] [output] [option]

#such as: python DetSim_WCDA_1.py /eos/user/z/zham/dcorsika/p5.e12/DAT000832.part ./out -settingfile ../config/settings.conf

Example 11: WCDA simulation(second step)

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/Wcda2Sim

a new directory, Wcda2Sim, will be created with some codes inside

- cd Wcda2Sim/cmt

a requirements file is used to configure Wcda2Sim package

- cmt config

#config package according to requirements file

- make

#compile and build Wcda2Sim package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python DetSim_WCDA_2.py [input] [output] [option]

Example 12: WTCTA simulation

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Simulation/WFCTASim

a new directory, WFCTASim, will be created with some codes inside

- cd WFCTASim/cmt

a requirements file is used to configure WFCTASim package

- cmt config

#config package according to requirements file

- make

#compile and build WFCTASim package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python DetSim_WFCTA.py

Example 13: WCDA reconstruction(core position)

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Reconstruction/WCDArec

a new directory, WCDArec, will be created with some codes inside

- cd WCDArec/cmt

a requirements file is used to configure WCDArec package

- cmt config

#config package according to requirements file

- make

#compile and build WCDArec package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python Rec_WCDA.py

Example 14: KM2A reconstruction(core position, direction)

- cd path_to_workarea
- svn co http://svn.lhaaso.ihep.ac.cn/LodeStar/offline/trunk/Reconstruction/KM2Arec

a new directory, KM2Arec, will be created with some codes inside

- cd KM2Arec/cmt

a requirements file is used to configure KM2Arec package

- cmt config

#config package according to requirements file

- make

#compile and build KM2Arec package

- source setup.sh

#make the library usable

- cd ../share

a python script to config this job

- python Rec_KM2A.py