Design and Development of STCF Offline Software

Xingtao Huang
For the STCF Software Team
Shandong University

Dec.4-7, 2018
Joint Workshop of future tau-charm factory
Orsay, France
Outline

- Requirements
- Architecture Design
- SNiPER Framework
- Detector Geometry Description
- Event Data Model
- Current Status
- Summary
Super Tau-Charm Facility in China

- e+ e- Collider
- $E_{cm} = 2-7$ GeV
- $L = 0.5-1 \times 10^{35} \text{ cm}^{-2}\text{s}^{-1}$ @4 GeV
- Event Rates: $\sim 100 \times \text{BESIII}$

- General purpose Detector
  - PXD
  - MDC
  - PID
  - EMC
  - MUD
- Event size: larger than BESIII
Requirements of STCF offline software

- Procedure and requirements of STCF are similar with BESIII
- Many of BESIII software could be reused in STCF
- But event rates and event size much larger than BESIII
- Need faster and more powerful offline software system than BOSS, The BESIII Offline Software System.
  - Underlying Framework: Gaudi → SNiPER
  - Detector Simulation
  - Event Data Model
  - Event Data Management System
  - IO system
  - Parallelism
  - …..
**OSCAR: Offline Software of Super Tau-Charm Facility**

- **ExLibs**: include frequently used third-party software and tools
- **SNiPER**: a new framework to provide core functionalities and common services.
- **Offline**: all software specific to STCF, including Generator, Simulation, Calibration, Reconstruction and Analysis.
Software Environments

- **Operation System and main Libraries:**
  - Scientific Linux 6.10
  - Gcc 4.9.4
  - ROOT 6.10.08
  - Geant4.10.04.p01

- **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 compile packages easily
  - Help users to setup the environment for running the application easily

- **Codes Management:** SVN repository
  - Keep the history of code evolution
  - Synchronization and sharing between developers
  - Tag and release
SNiPER Framework

- Developed by JUNO Collaboration since 2013
- Main goals:
  - lightweight framework, less dependence on third-party software
  - Flexible and fast execution
  - Easy to learn and convenient to use
  - Support multithreading
- Design and development
  - Learn a lot from other software frameworks, such as Gaudi
  - Based on the valuable experiences of DayaBay and BESIII
  - Coding from scratch
- Current Status
  - Successfully used for JUNO and LHAASO Experiments
  - Tested with BESIII Real Data Analysis
  - Under investigation by several other experiments, such as nXEO, CEPC…
  - We Started for STCF since Aug. 2018
Key Functionality of SNiPER

- **Dynamically Loadable Element**
  - Algorithm
  - Service
  - Task
- **Data Store**
- **Parallelism**
- **Property**
- **Logging**
Dynamically Loadable Element

- **Algorithm**
  - An unit of codes for Data Processing
  - Perform calculations during event loop
  - Users usually write their own algorithms
    - Inherit from the AlgBase

- **Service**
  - Similar with Algorithm
  - A piece of code for common use
    - GeometrySvc
    - DatabaseSvc
    - ……

- **Task**
  - AlgBase
    - initialize()
    - execute()
    - finalize()
  - SvcBase
    - initialize()
    - finalize()
  - Task
    - addAlg(name, scope)
    - addSvc(name, scope)
    - addTask(name, scope)
Task: A lightweight Application Manager

- Task manages its algorithms, services and sub-tasks

- All DLEs are organized in a tree structure

- Task controls the execution of algorithms

- One job can has more than one Tasks
  - SubTask provides nested event loop
  - Natral interface for multithread Computing (run each task in an individual thread)
Data Store for Event Data Management

- Data Store is the dynamically allocated memory place to hold events data which are being processed.
- Applications (in terms of algorithms) get events data from the Data Store and add/update objects after processing.
- Data type of DataStore is flexible.
Parallelism

- Developed based on Intel TBB to Support event level parallelism
  - Muster: Multiple SNiPER Task Scheduler
  - SniperTbbTask: Binding of a SNiPER Task to a TBB task
- Global DataStore to provide events for multi-tasks (or multi-threads)
- A dedicated task(thread) is used to read/write event data from/to files
◆ Configurable variable at run time

◆ Declare a property in DLElement(Alg, Svc, Tool and Task)

```cpp
//suppose m_str is a string data member
declProp("MyString", m_str);
```

◆ Configure a property in Python script

```python
alg.property("MyString").set("string value")
```

◆ Types can be declared as properties:
  ➔ scalar: C++ build in types and std::string
  ➔ std::vector with scalar element type
  ➔ std::map with scalar key type and scalar value type
SniperLog: a simple log mechanism supports different output levels

0: LogTest
2: LogDebug
3: LogInfo
4: LogWarn
5: LogError
6: LogFatal

```
LogDebug << "A debug message" << std::endl;
LogInfo << "An info message" << std::endl;
LogError << "An error message" << std::endl;
```

```
aHelloAlg.execute     DEBUG: A debug message
aHelloAlg.execute     INFO: An info message
aHelloAlg.execute     ERROR: An error message
```

Each DLElement(Alg,Svc,Tool, Task) has its own LogLevel and can be set at run time

⇒ very helpful for debugging

The output message includes more information, such as

⇒ where it happens
⇒ message level
⇒ message contents
More specific to Experiments

- **SNiPER provides the core functionalities and services**
  - User interfaces: Algorithm and Service
  - Task for management of event loop
  - DataStore for management of events in memory
  - Parallelisam for multithreading
  - Property
  - Logging

- **But without covering**
  - Detector Simulation
  - Event Data Model
  - File Input/Output
  - Others more specific to the experiments
DD4hep

- A general detector (geometry) description toolkit for HEP
- Developed in AIDA and AIDA2020
- Used by ILC, CLIC, FCC
- Under Evaluation by EIC, CMS and LHCb for upgrade

Key functionalities

⇒ Full Detector Description
  - Includes geometry, materials, visualization, readout, alignment, calibration, etc.

⇒ Full Experiment life cycle
  - From concept development, detector optimization, construction to operation.

⇒ Consistent Description with single source
  - for simulation, reconstruction, analysis, etc.

⇒ Ease of Use

Dec.06.2018
DD4hep Uses Root TGeo for the underlying geometry description
  ➔ Access to ROOT Open GL viewer for geometry
  ➔ Debugging of geometry

Default geometry description
  ➔ Compact xml-files and C++ drivers

Output formats/interface
  ➔ GDML
  ➔ Geant4 geometry
Visualization of STCF Detector with DD4hep

From Dong Liu

<detectors>
  <comment>Trackers</comment>

  <detector name="AirTube" type="AirTube" vis="VXDVis" id="42"
    insideTrackingVolume="true">
    <dimensions rmin="10.*mm" rmax="11.*mm" zhalf="6.250000000e+01*mm"/>
  </detector>

  ...

</detectors>
Event Data Model

- Defines the data unit to be processed in offline data processing
- In form of C++ classes
- Can be converted into persistent type and saved into disk

Data Processing

Generator  Simulation  Reconstruction  Analysis

DataStore (Memory)  File
Design of STCF EDM

- Single definition for Events in DataStore and Root Files
- Event Objects will be based on ROOT Tobjec to take advantages of ROOT powerful functionalities
  - Objects streamers
  - Class schema evolution
  - Run time type identification
  - Inspection, drawing etc.
- For each data processing stage, the event data is divided into two layers:
  - Header Object
  - Event Object
- IO system based on ROOT streamers supports data loading on demand to reduce I/O burden
Design Schema of STCF EDM
XML Object Description (XOD) Tool

- Traditionally writing C++ Code manually
  - Many repeatable work such as Getters and Setters
  - Difficult to be maintained
- Use XML file to define EDM
  - Strong syntax (DTD, XML Schema)
  - More readable, easier to maintain
  - Automatically generate the Get-, Set-functions, ROOT I/O Streamers
- A tool, XmlObjDesc (XOD), is developed to automatically generate class codes

*xml  XOD  *.h  CINT/CLing  *Dict.h *Dict.cc  G++  lib*.so

*xml  *.cc  *LinkDef.h

Dec.06,2018
The first working version of OSCAR has been installed in USTC computing nodes
- stcf01.ustc.edu.cn
- stcf01.ustc.edu.cn

Installation
- Automatic installation of the whole offline software with a shell script

Documentation
- OSCAR User Guide
  - http://cicpi.ustc.edu.cn/indico/getFile.py/access?contribId=1&resId=0&materialId=slides&confId=1610
- DD4HEP User guide
  - http://cicpi.ustc.edu.cn/indico/getFile.py/access?contribId=0&resId=2&materialId=slides&confId=1610

SVN repository
- Account request, please send mail to Wenhao Huang (whyellow@mail.sdu.edu.cn)
Summary

- OSCAR, the STCF offline software, is being designed and developed
  - Based on SNiPER and DD4hep
- First working version has been installed and several tutorials have been organized to train more people
- Lots of works ahead, more people are welcome
  - Optimization of Detector description
  - Integration DD4hep with OSCAR
  - Adding frequently used generators
  - Detector response simulation
  - Finalize Event Data Model
  - Design of File Input/Output System
  - Developing Reconstruction methods
- The goal: setup a full chain from generator to reconstruction for optimization of Detector design and performance study.
Thanks for your attention!