CEPCSW Prototype

Ziyan Deng, Xingtao Huang, Gang Li, Weidong Li, Tao Lin,
Shengsen Sun, Xiaomei Zhang, Jiaheng Zou

lintao@ihep.ac.cn

CEPC workshop
University of Oxford
15-17 April 2019
During R&D stage, CEPC seeks a lightweight data processing system. CEPCSW prototype is proposed in this talk.

- Motivation
- Requirements analysis
- CEPCSW prototype
  - Integration with Simulation
  - Migration of algorithms
- Timeline and manpower
- Summary and plans
Motivation

CEPC data volume and computing challenges

<table>
<thead>
<tr>
<th>Short term</th>
<th>Estimated Data Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>R&amp;D: detector design</td>
<td>~PB/year</td>
</tr>
<tr>
<td>Higgs/W factory</td>
<td>1.5~3 PB/year</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Long term</th>
<th>Estimated Data Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z factory</td>
<td>0.5~5 EB/year challenge</td>
</tr>
</tbody>
</table>

Requirements:

- R&D stage: fast development and iteration.
- Operation: extensible for new technologies.
Data processing chain

CEPC Detector

Simulation

HLT #

Analysis Data

Rec Event Data

Raw Event Data

Conditions

Event Data & Non-Event Data

Analysis & Event Display

Pre-Selection

Reconstruction

Alignment & Calibration

# HLT: High Level Trigger
Software stack

CEPCSW prototype provides flexible integration of the applications.
<table>
<thead>
<tr>
<th>General</th>
<th>Features</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Modern software engineering</td>
<td>object-oriented, modular, configurable, load dynamically.</td>
</tr>
<tr>
<td></td>
<td><strong>Lightweight &amp; compact</strong></td>
<td>Less dependencies, Portable, Easily maintained</td>
</tr>
<tr>
<td>Extensible &amp; simple</td>
<td>Extending functionalities easily.</td>
<td></td>
</tr>
<tr>
<td>Workflow management</td>
<td>Flexible, event loop &amp; event filter</td>
<td></td>
</tr>
<tr>
<td>Data management</td>
<td>Thread-safe transient object management in memory.</td>
<td>Support data from disk, tape and network.</td>
</tr>
<tr>
<td></td>
<td>Algorithms developed in offline can be used online.</td>
<td></td>
</tr>
<tr>
<td>Application domain</td>
<td><strong>Detector performance study</strong></td>
<td>Comparison of multiple detector designs.</td>
</tr>
<tr>
<td></td>
<td>Physics performance study</td>
<td>Algorithm development and physics pre-study.</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
<td>full simulation, fast simulation, etc.</td>
</tr>
<tr>
<td></td>
<td>Reconstruction</td>
<td>Migration of other algorithms.</td>
</tr>
<tr>
<td>New technologies</td>
<td><strong>Parallel computing</strong></td>
<td>Multithreading, MPI, GPU</td>
</tr>
<tr>
<td></td>
<td>Machine learning</td>
<td>Tensorflow</td>
</tr>
<tr>
<td></td>
<td>Big data</td>
<td>spark and dataframe</td>
</tr>
<tr>
<td></td>
<td>Supercomputers</td>
<td>Specific computing architectures</td>
</tr>
</tbody>
</table>
Event Data Processing Framework

• Modern frameworks share the same concepts.

• Framework is adopted by experiments according to their requirements.
  • Gaudi: FCC, LHCb, ATLAS
  • Marlin: ILC
Gaudi

- Developed by LHCb, became CERN standalone project.
- BESIII and Daya Bay experiments used Gaudi.
- Good design, but complicated.
- **Requires a dedicated expert team to maintain.**
Marlin

- Developed by ILC, used for Reconstruction & Analysis.
- A simple framework based on LCIO.
- Only used in R&D.

Diagram:
- Marlin::main
  - Processor0
  - Processor1
  - Processor2
  - ProcessorN
  - OutputProcessor

 Files:
- MyInput2.slcio
- MyInput1.slcio
- MyInput0.slcio

Event:
- LCEvent
  - collection0

Collections:
- Read and add collections

Output:
- MyOutput.slcio
Framework comparison

<table>
<thead>
<tr>
<th></th>
<th>Gaudi</th>
<th>Marlin</th>
<th>CEPC intended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainer</td>
<td>CERN EP-SFT</td>
<td>DESY IT</td>
<td>CEPC dedicated team</td>
</tr>
<tr>
<td>Experiments</td>
<td>FCC, LHCb, etc.</td>
<td>ILC, CEPC</td>
<td>CEPC</td>
</tr>
<tr>
<td>Functionalities</td>
<td>Complex, Heavy</td>
<td>Simple</td>
<td>Lightweight, Extensible</td>
</tr>
<tr>
<td>Event Data Model</td>
<td>Gaudi::DataObject</td>
<td>LCIO</td>
<td>ROOT</td>
</tr>
<tr>
<td>External Libraries</td>
<td>~35 of LCG (~100 pkgs)</td>
<td>Lightweight (~10 pkgs)</td>
<td>Lightweight (~10 pkgs)</td>
</tr>
<tr>
<td>API &amp; Configuration</td>
<td>C++, Python</td>
<td>C++, XML</td>
<td>C++, Python</td>
</tr>
<tr>
<td>Parallel Computing</td>
<td>GaudiHive, Algorithm-</td>
<td>Not supported</td>
<td>Multiple levels (node, event, intra-event)</td>
</tr>
<tr>
<td></td>
<td>level</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- CEPC seeks a lightweight framework to fulfill the requirements from R&D and data processing.
  - Create a new one or develop based on existing?
SNiPER

- A lightweight framework, developed by IHEP & SDU (~2013)
- Use ROOT as transient and persistent data model.
- Scalable: multi-threading support (~2016)

A simple framework with key Gaudi design features.

- **Flexible workflow.** (Event Filter & Event Mixing)
- **Parallel computing**

Task: similar to Gaudi application manager (event loop), manages algorithms, services and sub-tasks. Easy to be parallelized.

- **No Converters.** Straightforward mapping between transient ROOT objects and persistent data in ROOT files.
- **Thread-safe event data management**

Plan: develop a CEPCSW prototype based on existing code.
CEPCS W Prototype Tasks

Develop components:

• ROOT-based Event Data Model & I/O
  • Compatible with LCIO EDM.

• LCIO compatible reader
  • During the migration, reading existing simulated samples.

• Unified geometry system for Sim/Rec/A na
  • Evaluate DD4hep

Also integrate and migrate existing Sim/Rec/A na algorithms.
Integration with DD4hep

- Service providing **unified access to geometry** via DD4hep.
- Simulation algorithms produce sim events into ROOT files.
Migration of Rec/Ama algorithms

- Easy to translate algorithms between frameworks.

<table>
<thead>
<tr>
<th></th>
<th>Gaudi</th>
<th>Marlin</th>
<th>CEPCSW prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Algorithm</td>
<td>Processor</td>
<td>Algorithm</td>
</tr>
<tr>
<td>Base class name</td>
<td>Algorithm</td>
<td>marlin::Processor</td>
<td>AlgBase</td>
</tr>
<tr>
<td>initialization</td>
<td>initialize()</td>
<td>init()</td>
<td>initialize()</td>
</tr>
<tr>
<td>Execution during event loop</td>
<td>execute()</td>
<td>processEvent(LCEvent*)</td>
<td>execute()</td>
</tr>
<tr>
<td>finalization</td>
<td>finalize()</td>
<td>end()</td>
<td>finalize()</td>
</tr>
<tr>
<td>Data access</td>
<td>Event Data service</td>
<td>LCIO</td>
<td>Event Data service</td>
</tr>
<tr>
<td>Service support</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Tool support?</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Accessing event data

**Marlin**

```cpp
void MyAnalysisProcessor::processEvent( LCEvent * evt ) {
    LCCollection * col = evt->getCollection(trackColName);
    int nelem = col->getNumberOfElements();
    for (int i=0;i<nelem; ++i) {
        Track * trk = dynamic_cast<Track*>(col->getElementAt(i));
    }
    // ...
}
```

**Gaudi/CEPCSW prototype**

**Consistent track interface**

```cpp
bool MyAnalysisAlg::execute() {
    SmartDataPtr<Event> evt( eventSvc(), "/Event/Header" );
    SmartDataPtr<MyTrackVector> myTracks(eventSvc(), "/Event/MyTracks" );
    int nelem = myTracks->size();
    for (int i=0;i<nelem; ++i) {
        MyTrack* trk = (*myTracks)[i];
    }
    // ...
    return true;
}
```
## CEPCS W Timeline & Manpower

<table>
<thead>
<tr>
<th>Date</th>
<th>Component</th>
<th>Task</th>
<th>Manpower</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018.11</td>
<td>General</td>
<td>Design, requirements discussion</td>
<td>Jiaheng, Xingtao, Ziyan, Tao</td>
</tr>
<tr>
<td></td>
<td>Core+Sim</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019.04</td>
<td>General</td>
<td>This workshop (prototype proposal)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Core</td>
<td>Event Data Model &amp; ROOT I/O</td>
<td>Jiaheng, Xingtao</td>
</tr>
<tr>
<td>2019.05</td>
<td>Sim</td>
<td>Integration with DD4hep</td>
<td>Ziyan, Tao</td>
</tr>
<tr>
<td>2019.06</td>
<td>Release</td>
<td>Initial prototype (Core+Sim)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>General</td>
<td>Tutorial &amp; Documentation</td>
<td></td>
</tr>
<tr>
<td>2019.07</td>
<td>Rec</td>
<td>Migration of algorithms</td>
<td>Chengdong, Yao</td>
</tr>
<tr>
<td>2019.11</td>
<td></td>
<td>Demonstration with sim+track rec</td>
<td></td>
</tr>
</tbody>
</table>

Welcome to join the team!
Summary and plans

• A well designed data processing chain is important in short term (R&D) and long term (Operation).

• To fulfill the requirements, a CEPCSW prototype is proposed.
  • Developing based on an existing lightweight framework.
  • Easy to migrate and reuse existing algorithms.
  • Three components to be developed (EDM & ROOT I/O).

• Plans:
  • June 2019: initial prototype with core software and simulation.
  • Nov 2019: demo with simulation and track reconstruction.
Thank you!