Progress of online process software framework for CEPC ref-detector

Xu Zhang

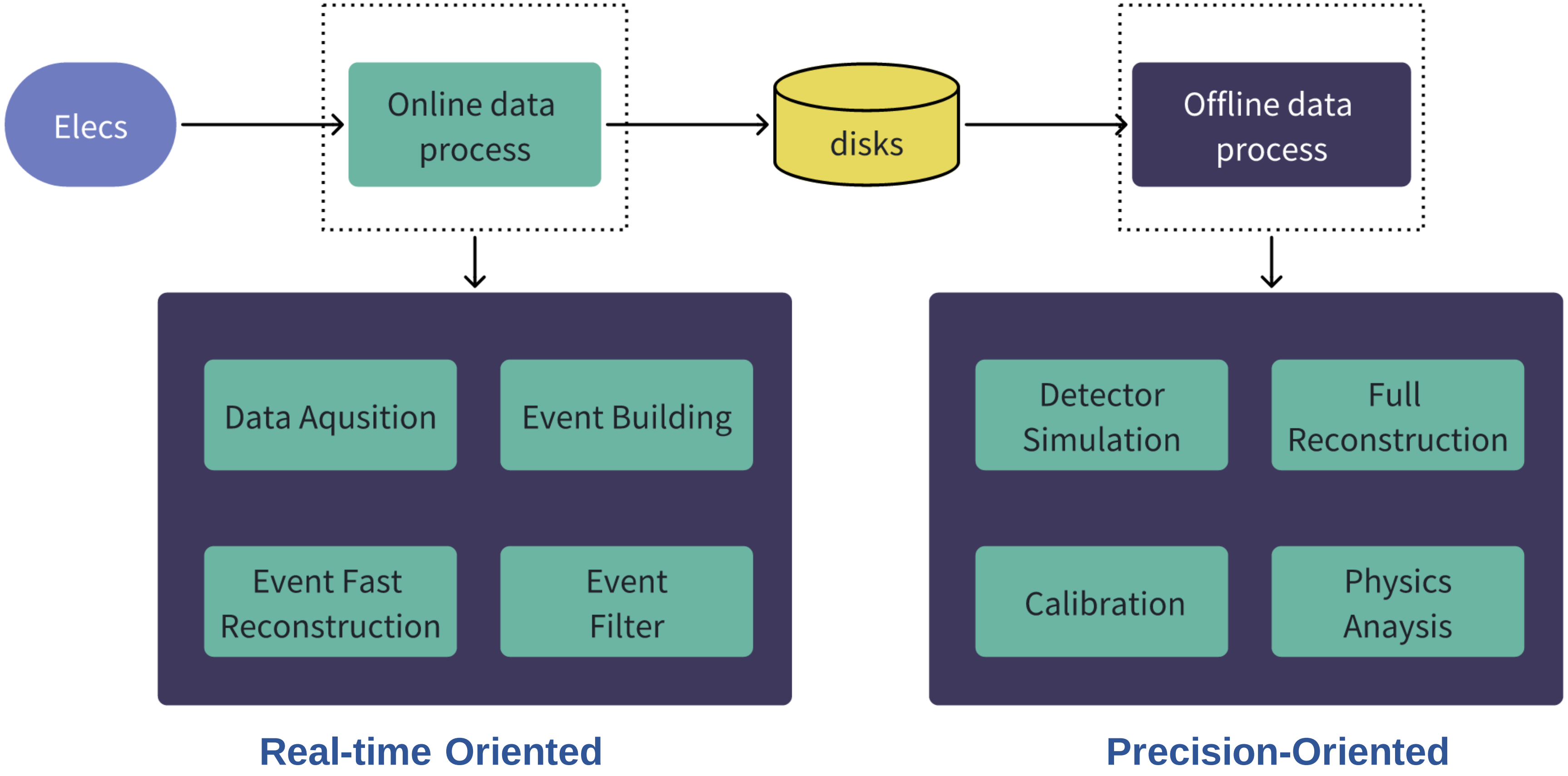
On behalf of CEPC TDAQ GROUP

High energy physics data process

HEP experiment data pross

- Online: Fast reconstruction and event selection for data volume reduction on disk.
- Offline: Calibration and full reconstruction for physical analysis.

Online algorithms can originate from offline algorithms.



Online software framework: Radar

heteRogeneous Architecture of Data Acquisition and pRocessing

Radar is a well-tested online software framework: It has been implemented on LHAASO(RadarV1.0) and JUNO(RadarV2.0).

Radar can handle different data acquisition modes: Including hardware-triggered and triggerless readout.

LHAASO

Large High Altitude Air Shower Observatory

Scientific Goals:

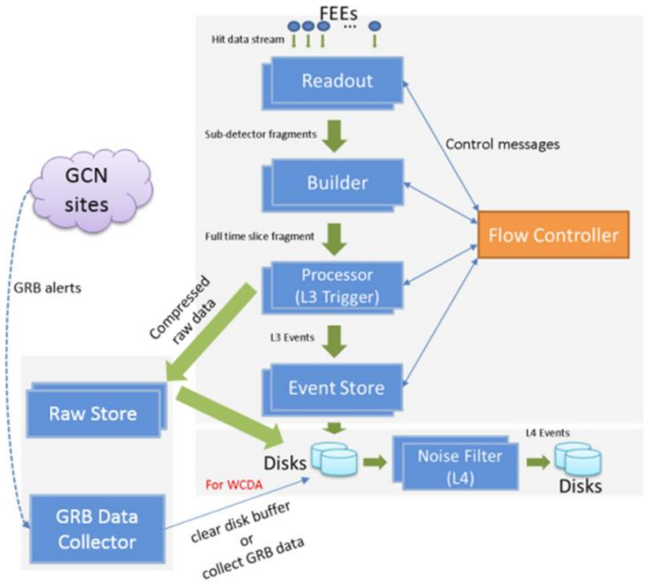
- Explore the origin of high-energy cosmic rays and conduct fundamental research on related high-energy radiation, astrophysical evolution, and dark matter distribution.

Experimental Facilities:

- KM2A: A 1.3-square-kilometer detector array comprising 5,195 EDs and 1,171 MDs.
- WCDA: Three water tanks equipped with 3,120 PMTs including 900 eight-inch and 2,220 twenty-inch tubes.
- WFCTA: 20 telescopes.

Data Volume:

- Readout data rate: 5 GB/s
- Stored data volume: 300 MB/s



JUNO

Jiangmen Underground Neutrino Observatory

Scientific Goals:

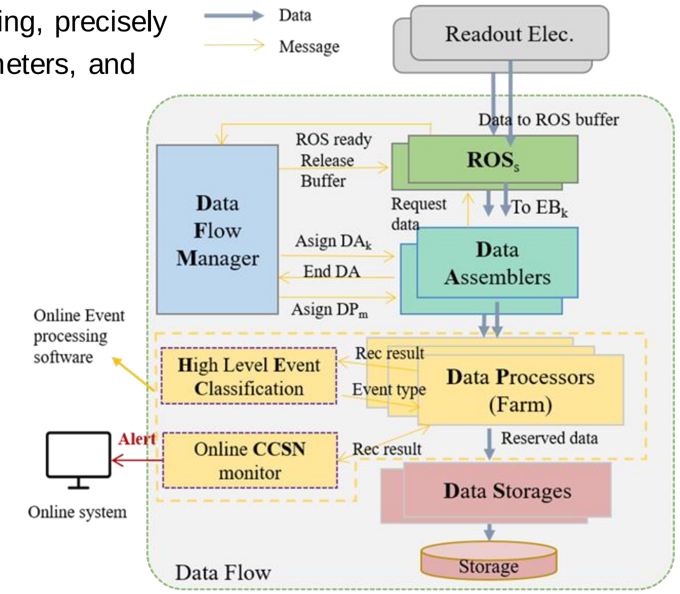
- Determine neutrino mass ordering, precisely measure neutrino mixing parameters, and observe supernovae, etc.

Experimental Facilities:

- CD LPMT: 17612
- CD SPMT: 25600
- WP LPMT: 2400
- Top Tracker

Data Volume:

- Readout data rate: 40 GB/s
- Stored data volume: 60 MB/s

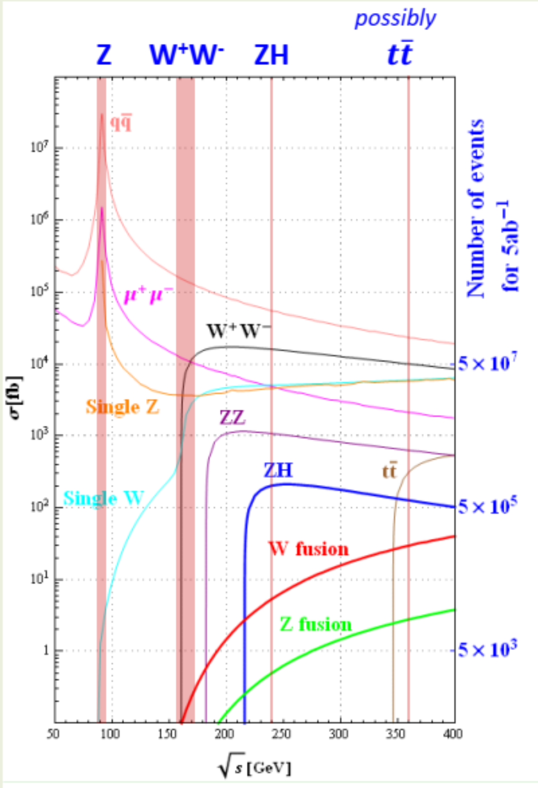


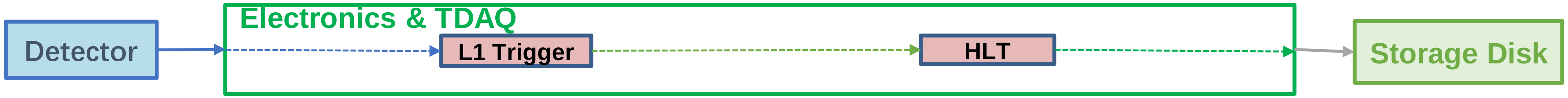
CEPC ref-Detector Requirement

CEPC ref-Detector TDAQ System requirement:

- Higgs: 1.3MHz -> 1kHz
- Low Z: 12MHz -> 20kHz
- High Z: 39.4MHz -> 154kHz

Running mode SR power	Higgs 50 MW	Z 12.1 MW
Non-empty bunch crossing rate(MHz)	1.34	12
Luminosity ($10^{34}/cm^2/s$)	8.3	26
Physical event rate (kHz)	0.5	10
L1 triger rate (kHz)	20	120
DAQ readout rate (Gbyte/s)	5.34	11.9
HLT rate (kHz)	1	20
Raw event size (kbyte)	405	333
DAQ storage rate (Gbyte/s)	0.405	6.66





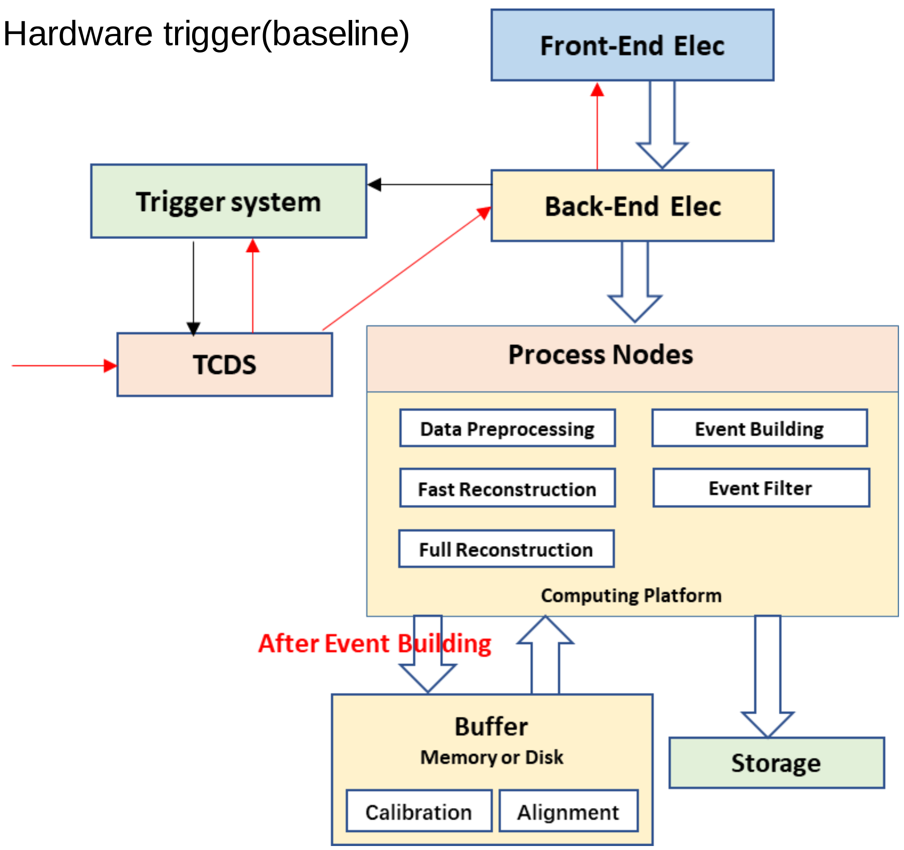
Bunch crossing rate	After L1 trigger rate	After HLT rate
Higgs mode: 1.3 MHz (130 GB/s)	50 kHz(6 GB/s)	1 kHz(0.4 GB/s)
Low Lumi. Z: 12 MHz (220 GB/s)	120 kHz(12 GB/s)	20 kHz(6.7 GB/s)
High Lumi. Z: 39.4 MHz (~1300 GB/s)	400 kHz(~105 GB/s)	154 kHz(~67 GB/s)

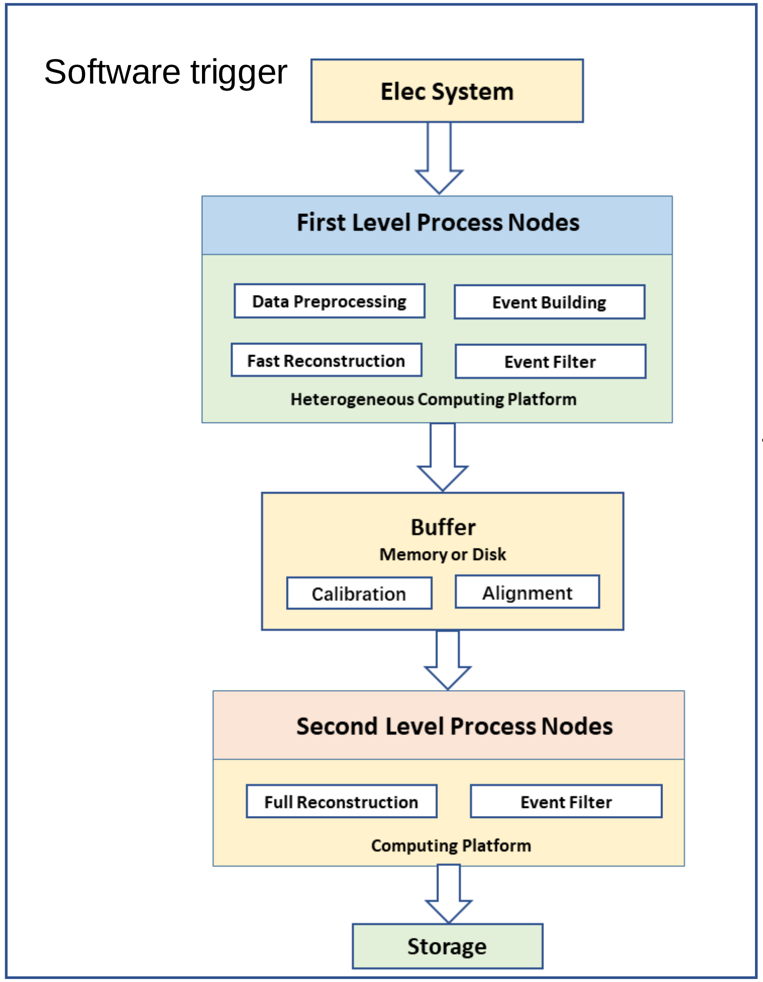
Readout Strategy

CEPC ref-Detector readout strategy:

Hardware trigger(baseline): Fast speed, low cost. Lower online software burden: 400k(105GB/s) -> 154k(67GB/s)@High Z

Full Software trigger: High flexibility. **Higher online software burden:** 39.3MHz(1300GB/s) -> 1k(67GB/s)@High Z





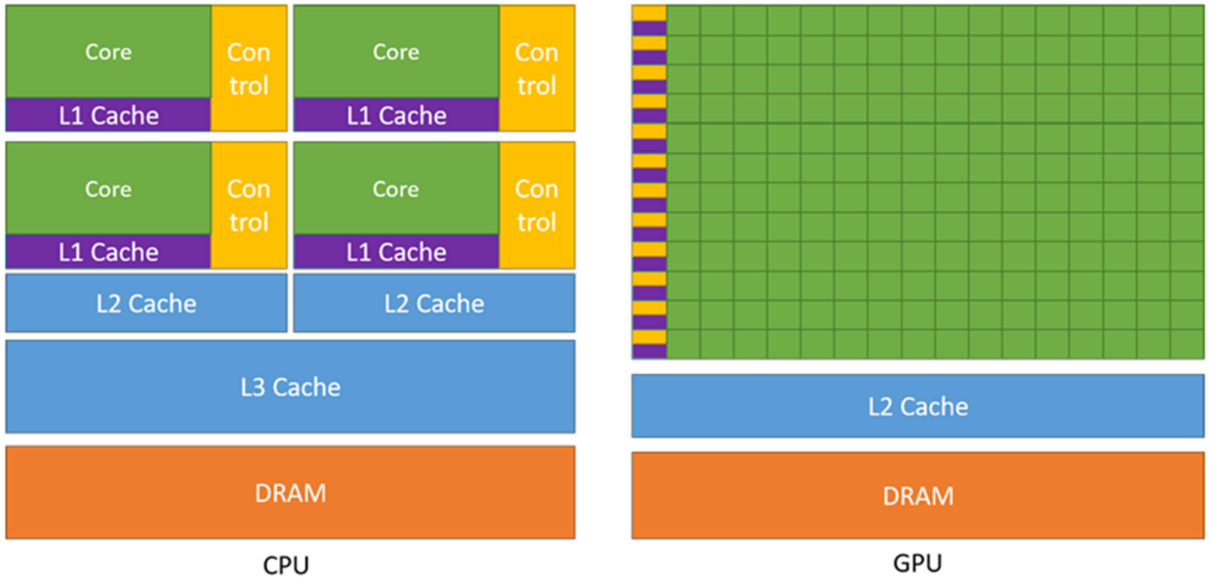
This talk focuses on this situation.

Heterogeneous Computing

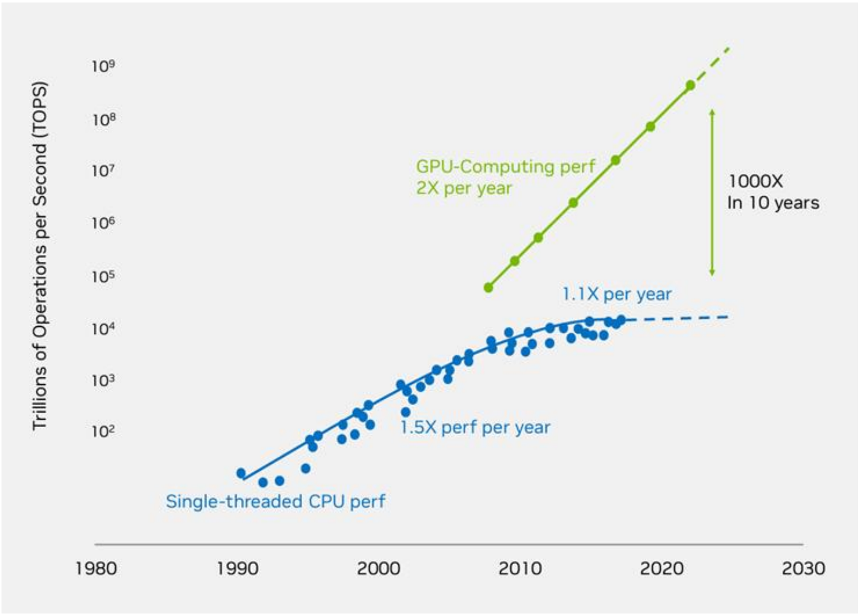
Heterogeneous computing: Enhance performance by using different types of processors for their respective tasks.

Collider Adaptation: Online algorithms in collider experiments are well-suited for parallel computing.
Proven Feasibility: Heterogeneous computing is already deployed in the online process of LHCb & ALICE.
Bright Prospects: 1000x GPU performance gain in a decade.

Heterogeneous computing enables the RADAR framework to achieve higher performance.



CUDAC++ProgrammingGuide

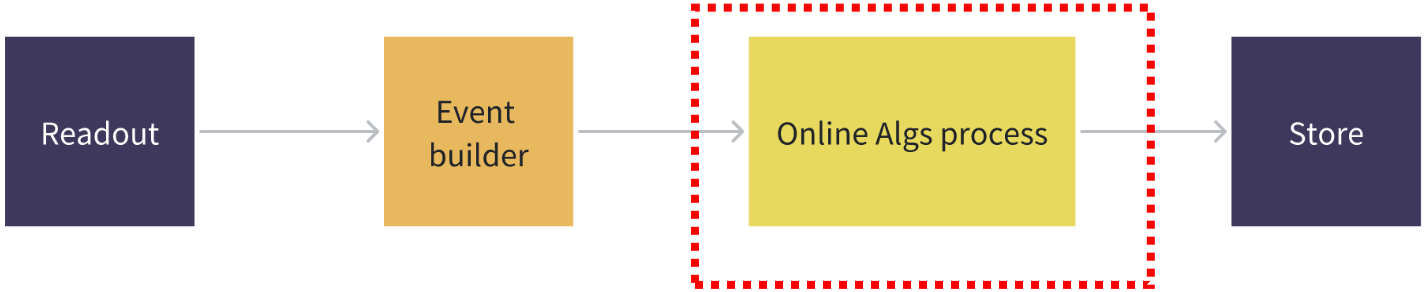


NVIDIA Investor Presentation October 2022

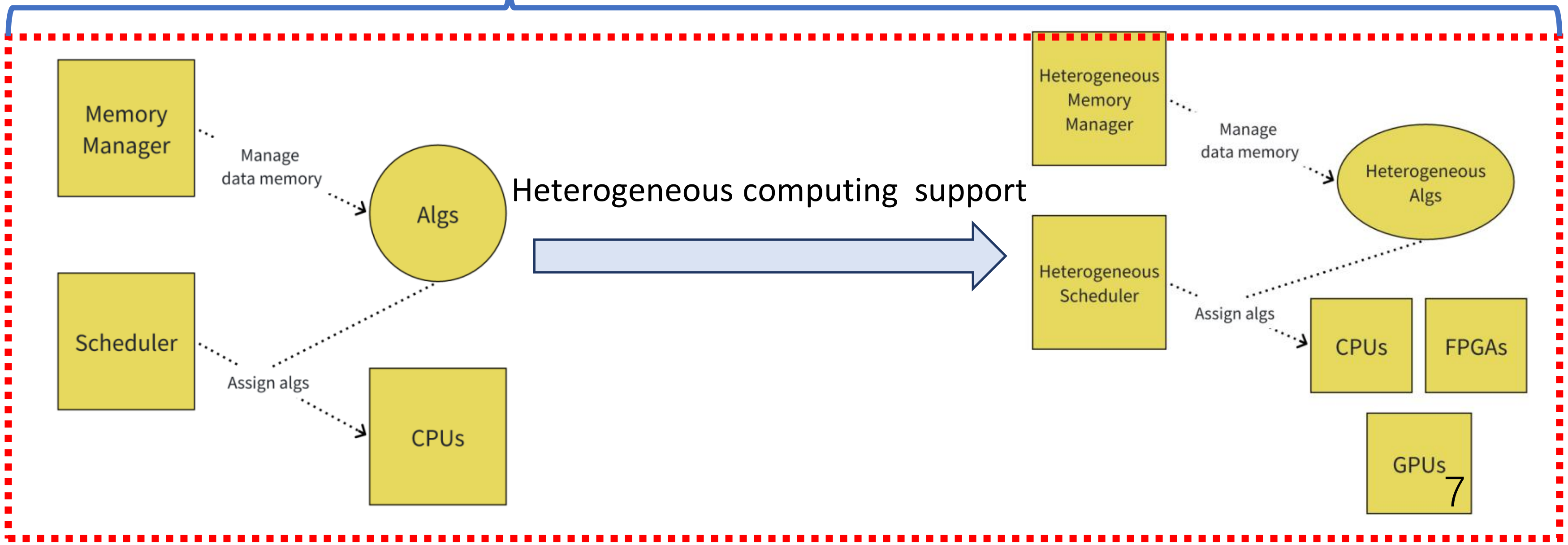
RadarV3.0 upgrade: heterogenous computing

RADAR framework:

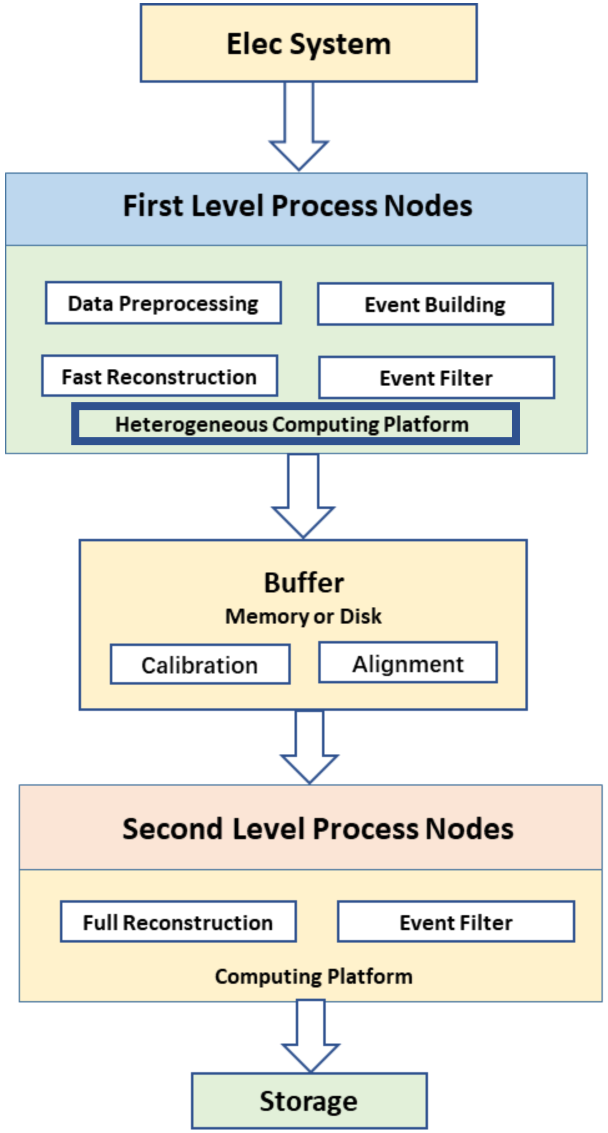
- **Readout:** Transferring data from electronics.
- **Builder:** Combining partial detector data into full detector data.
- **Online Algs:** Fast reconstruction for reducing data.
- **Store:** Storing the data on disk.



RADAR V3.0: TB/s-scale data processing



Upgrade Focus: Scheduler and Memory Manager



Scheduler research:

Goal: Minimize memory transfer across devices.
Approach: Schedule dependent algorithms on the same device to enable direct data reuse.

Memory manager research:

Goal: Efficient Management of Heterogeneous Memory Resources.
Approach: Using heterogeneous memory pool.

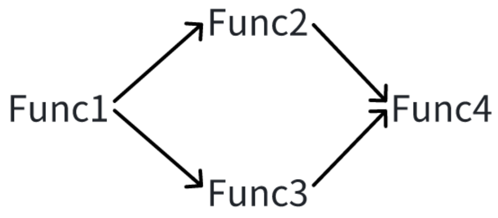
Goal: Simplify
Approach: Using a unified data view across all devices.

Scheduler Implement

1. The Framework generates a stream based on the inputs and outputs of the user's algorithm.

	Input	Output
Func1		func1_output
Func2	func1_output	func2_output
Func3	func1_output	func3_output
Func4	func2_output func3_output	func4_output

a. Users define input and output of algs.



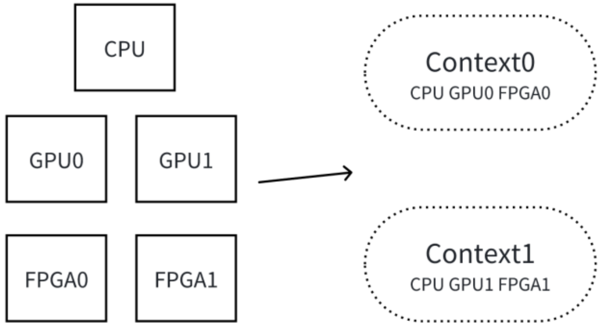
b. Framework generate task graph.

Stream:

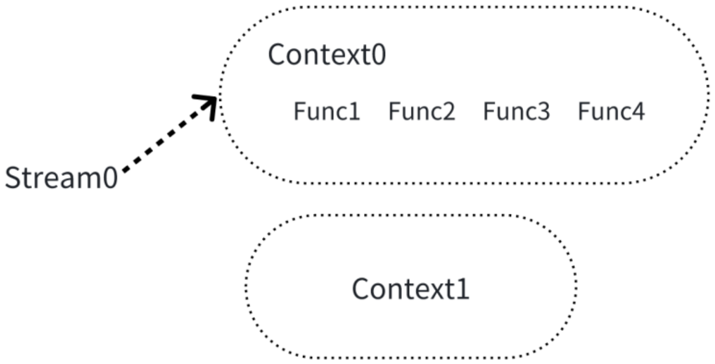
Func1 Func2 Func3 Func4

c. Generate a task stream in topological order.

2. The Framework packages a set of devices into a context, and the stream will be executed within a single context.

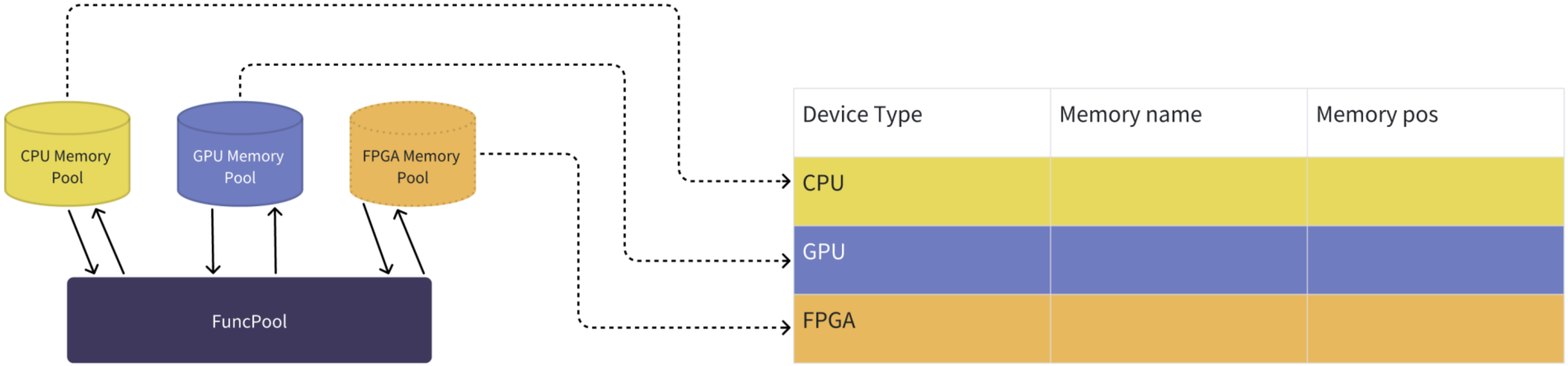


a. Generate context.



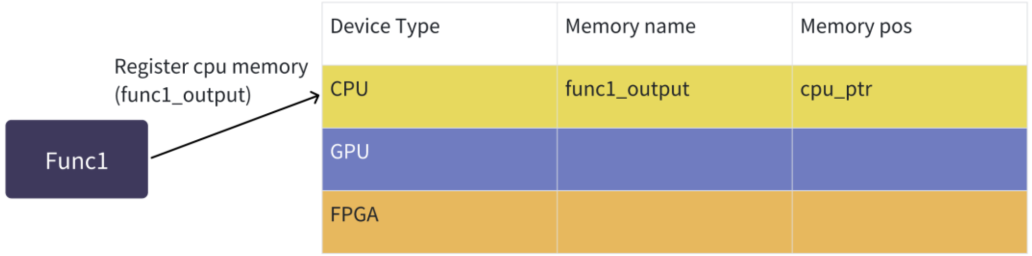
b .Dispatch a single stream to one context.

Memory Manager Implement



unified data view:
Users do not need to know on which device memory their dependent data resides.

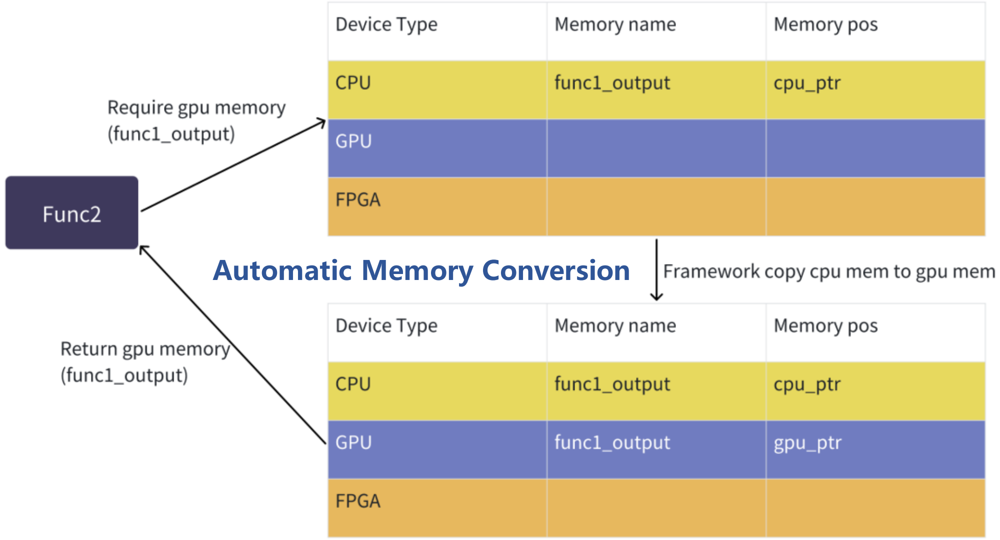
e.g., GPU func2 needs the output of CPU func1 as its input .



1.Func1 registers its output on map: func1_output.

2. GPU Func2 requires func1_output.

3.There is only CPU memory for func1_output and no GPU memory, the framework will automatically copy it to GPU memory for Func2 to use.

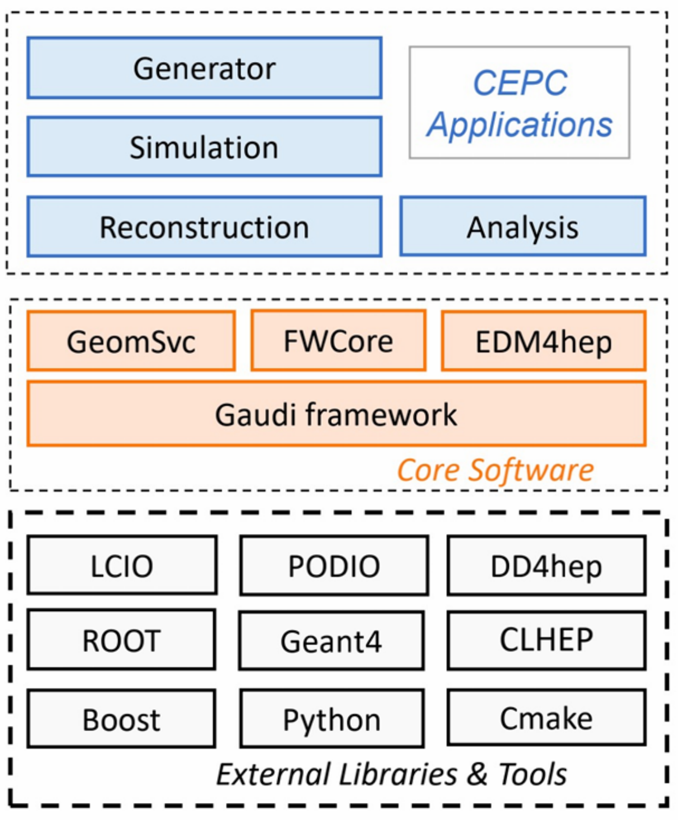


Algorithm integration research

Offline algorithm integration friendly:

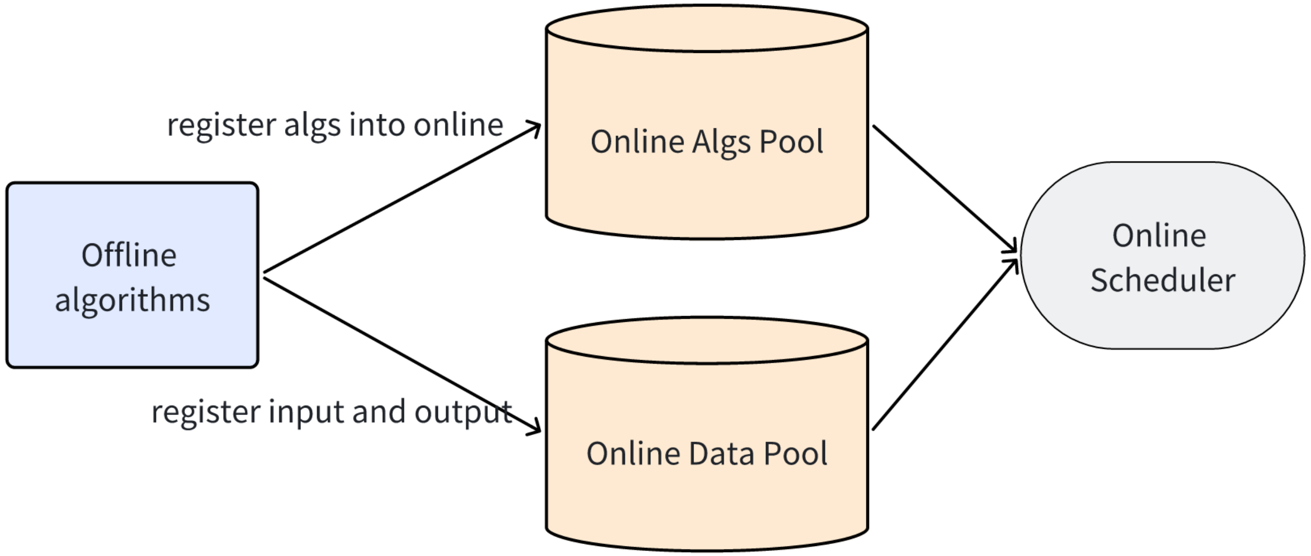
Introduced some designs to facilitate the integration of offline algorithms into online systems.

- Online framework integrates **EDM** and **service** modules for offline algorithm(**CEPCSW**).
- Variable names are kept consistent between online and offline data format headers.

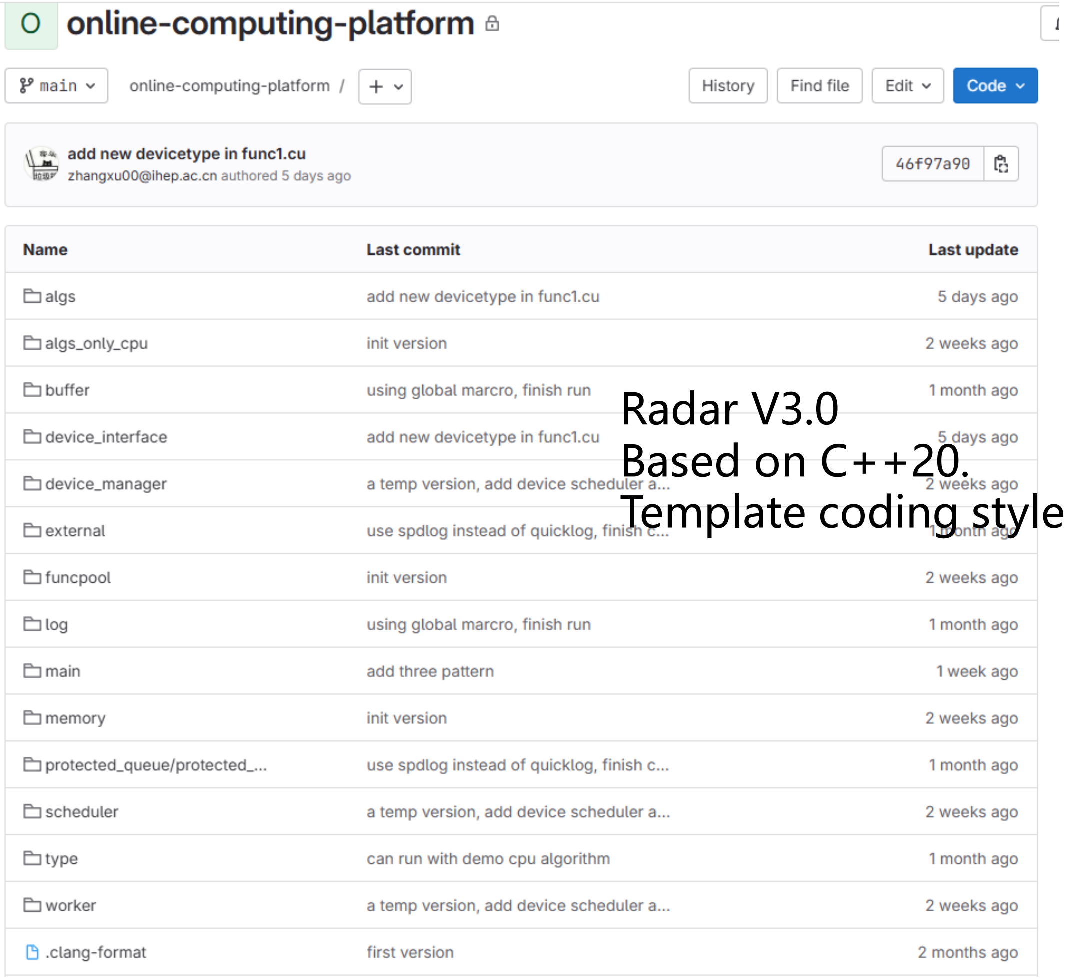


CEPCSW: the Key4hep-based software for CEPC

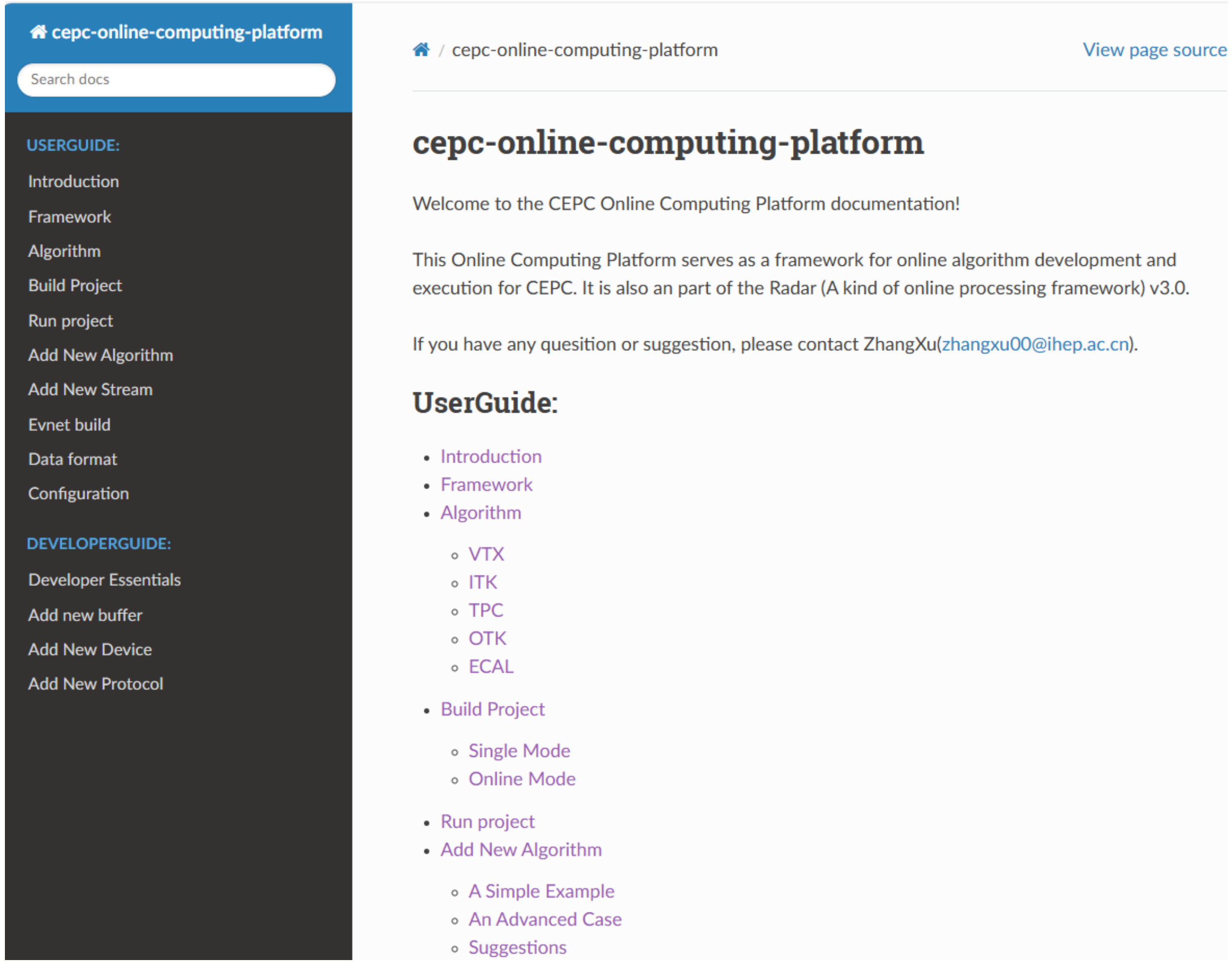
By changing the header and namespace,
offline algorithms can adapt to the online data format.



The online computing framework initial version is being tested.



Radar V3.0
Based on C++20.
Template coding style.



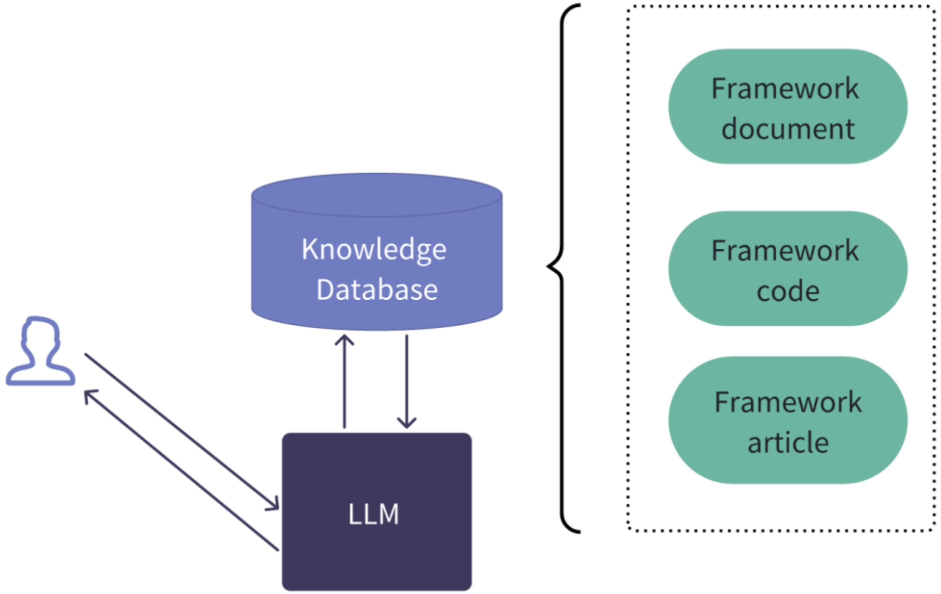
Tool provided for users

Developer friendly: Facilitates users learning and using the framework.

Deployed the local large language model helps users learn and understand the frameworks documentation.

By using the knowledge base, the large model provides better answers to questions about the **CEPC online framework**.

Use framework-related information to serve as the **knowledge base** for the large language model.



How to Add New Algorithm in this framework

The large model will generate responses based on knowledge database.

Okay, the user is asking "How to Add New Algorithm in this framework" after previously discussing how to run the RADAR V3.0 project. Let me analyze the context to provide a precise answer.

Looking at the provided context documents (contents.md with multiple contexts), I see detailed instructions about adding algorithms. The key points from Context 0 are:

01. Place new algorithm under `project/algs/`
02. Must define header and source files
03. Need to register with:
    - Algorithm class name
    - Stream (HLT1/HLT2)
    - Input/output (output must be a struct in `./type/type/datatype.h`)
    - Configuration path
    - Include `init()` and `run()` methods

Context 2 provides a concrete example with:

CEPC online process source documents

```c++
REGISTER_TASKTYPE(TaskType::kHlt1);
REGISTER_CLASS(Func1);
REGISTER_OUTPUT(Func1, Func1Output);
REGISTER_CONFIG(Func1, "config/func1_config.json");
REGISTER_DEVICETYPE(Func1, DeviceType::kCPU);
```

Context 3 discusses configuration details but isn't directly about adding algorithms.

The user's question is very specific about the framework's requirements for adding algorithms. I should provide the exact steps based on the documentation, not general advice.

I'll structure the answer to cover:

- File placement
- Required registration macros (from Context 2)
- Input/output struct definition
- Configuration setup
- Implementation of init()/run() methods

Summary

RADAR Framework Upgrade for TB/s-Scale Data Triggering

- **Core Enabler:** Heterogeneous Computing Architecture
- **Key Enhancements:**
  Heterogeneous Scheduling Service.
  Heterogeneous Memory Management.
- **Algorithm Integration:** Simplify the algorithm integration process for current framework test.
- **Next step:**

Type	Supported	To be supported
Buffer	Disk	Shared Memory • Memory Buffer
Protocol	TCP	RDMA
Compute	CPU • GPU	FPGA
Memory	CPU • GPU	FPGA

THANKS!