

# Application of TRACCC seeding to the CEPC vertex detector

YiZhou Zhang, WeiDong Li, Xiaocong Ai, Tao Lin  
[zhangyz@ihep.ac.cn](mailto:zhangyz@ihep.ac.cn)

The Institute of High Energy Physics, IHEP, China

CHEP2024  
Krakow, Poland  
23<sup>rd</sup> Oct 2024

# Outline

- 1 Introduction
- 2 Integration of TRACCC with CEPCSW
- 3 Geometry & EDM
- 4 Extension of seeding algorithm
- 5 Performance
- 6 Summary

The CEPC is a 100 km circular electron-positron collider aiming to

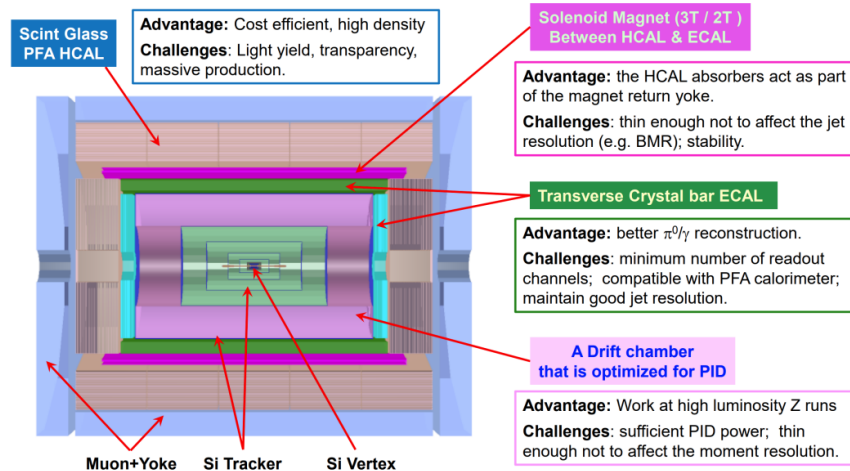
- Precisely measure the Higgs boson's properties
- Study electroweak physics at Z-boson peak
- Will produce:
  - At 250 GeV: Higgs bosons ( $4 \times 10^6$ )
  - At 160 GeV: W bosons ( $> 10^8$ )
  - At 90 GeV: Z bosons ( $> 4 \times 10^{12}$ )

The conceptual design report (CDR) has been completed in Oct. 2018.

- High track efficiency ( $\sim 100\%$ )
- High momentum resolution ( $\sim 0.1\%$ )

The 4th conceptual detector was proposed on the basis of the CEPC CDR

- is characterized by a combination of **silicon detectors** and **drift chamber** designed to provide both tracking and PID for charged particles



Schematic view of CEPC's 4th Concept Detector

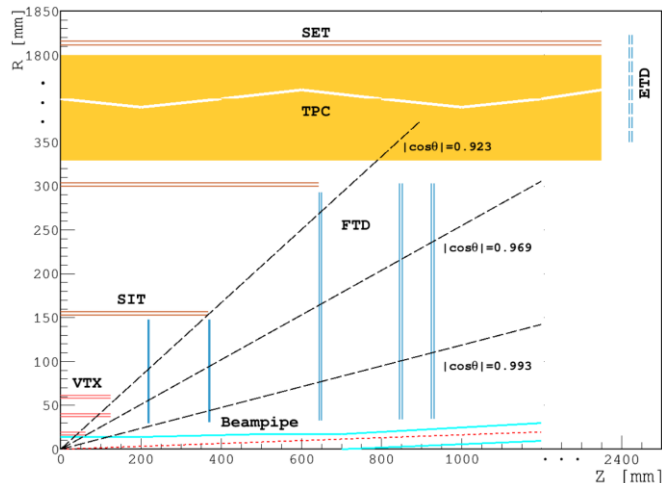


## CEPC Vertex Detector

- is the innermost tracker playing a dominant role in determining the vertices of a collision event.
- Covers:
  - radial range from 16 mm to 60 mm
  - Z range from -125mm to 125mm

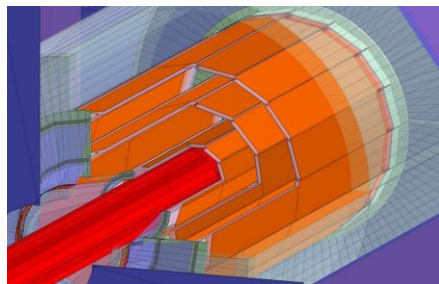
The baseline layout of the Vertex Detector consists of

- 6 concentric cylindrical layers of high spatial resolution silicon pixel sensors.
- Two layers of silicon pixel sensors are mounted on both sides of each of **three ladders** to provide **6** space points.



Layout of the CEPC baseline tracker

The VTX is located closest to the interaction point.



Schematic view of CEPC Vertex Detector

Only the silicon sensor sensitive region (in orange) is depicted.

The vertex detector surrounds the beam pipe (in red).

	$R$ (mm)	$ z $ (mm)	$ \cos\theta $	$\sigma$ ( $\mu\text{m}$ )
Layer 1	16	62.5	0.97	2.8
Layer 2	18	62.5	0.96	6
Layer 3	37	125.0	0.96	4
Layer 4	39	125.0	0.95	4
Layer 5	58	125.0	0.91	4
Layer 6	60	125.0	0.90	4

Layers of CEPC Vertex Detector

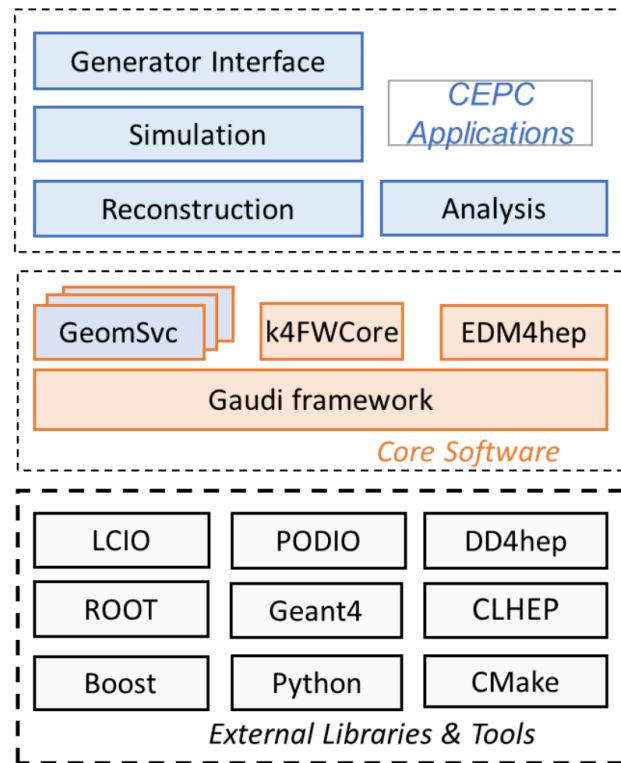


CEPCSW is organized as a multi-layer structure

- Applications: **simulation, reconstruction and analysis**
- Core software
- External libraries

The key components of core software include:

- **Gaudi/Gaudi Hive**: defines interfaces to all software components and controls their execution
- **EDM4hep**: generic event data model
- **k4FWCore**: manages the event data
- **DD4hep**: geometry description
- **CEPC-specific components**: GeomSvc, detector simulation, beam background mixing, fast simulation, machine learning interface, etc.



## TRACCC

- \* **TRACCC** is one of the ACTS R&D projects, which is developing full track reconstruction algorithms that can run on accelerators.
- has been introduced at Beomki's talk:
  - <https://indico.cern.ch/event/1338689/contributions/6010050/>
- Is standalone and features a modular architecture
- Has excellent physics and computing performance



Category	Algorithms	CPU	CUDA	SYCL	Alpaka	Kokkos	Futhark
Clusterization	CCL / FastSv / etc.	✓	✓	✓	●	○	✓
	Measurement creation	✓	✓	✓	●	○	✓
Seeding	Spacepoint formation	✓	✓	✓	●	○	○
	Spacepoint binning	✓	✓	✓	✓	✓	○
	Seed finding	✓	✓	✓	✓	○	○
	Track param estimation	✓	✓	✓	✓	○	○
Track finding	Combinatorial KF	✓	✓	●	●	○	○
Track fitting	KF	✓	✓	✓	○	○	○
Ambiguity resolution	Greedy resolver	✓	○	○	○	○	○

✓: exists, ●: work started, ○: work not started yet

## SYCL

- SYCL is a high-level C++ programming model. An uniformed written code can run on a variety of platforms.
- \* High Portability and Programming Efficiency 🖱️

Status of TRACCC

## Challenges for tracking in the CEPC Vertex Detector

### 1. Piling-up of multiple events

- The size of detector time window (117 pile-up for  $t\bar{t}$ ) is determined by DAQ

### 2. High beam-related background

- particularly in Z energy region

### 3. Reuse of offline tracking algorithm for the purpose of online high level trigger

- Same rec algorithm: offline environment & online Event Filter

#### Conclusion:

- huge # of hits & background / event:
  - high demand on track recognition
  - substantial computational load

→ heterogeneous computing (e.x. TRACCC) and parallelization techniques are required.

\* This contribution mainly focuses on:

- the implementation of seeding algorithm for the vertex detector (VTX), based on TRACCC, in the CEPCSW environment.

	Higgs	Z	W	$t\bar{t}$
SR power per beam (MW)	50			
Bunch number	446	13104	2162	58
Bunch spacing (ns)	346.2 ( $\times 15$ )	23.1 ( $\times 1$ )	138.5 ( $\times 6$ )	2700.0 ( $\times 117$ )
Train gap (%)	54	9	10	53
Luminosity per IP ( $10^{34} \text{ cm}^{-2} \text{ s}^{-1}$ )	8.3	192	26.7	0.8

Requirements: Physical Event Rate

# Outline

1

Introduction

2

Integration of TRACCC with CEPCSW

3

Geometry & EDM

4

Extension of seeding algorithm

5

Performance

6

Summary




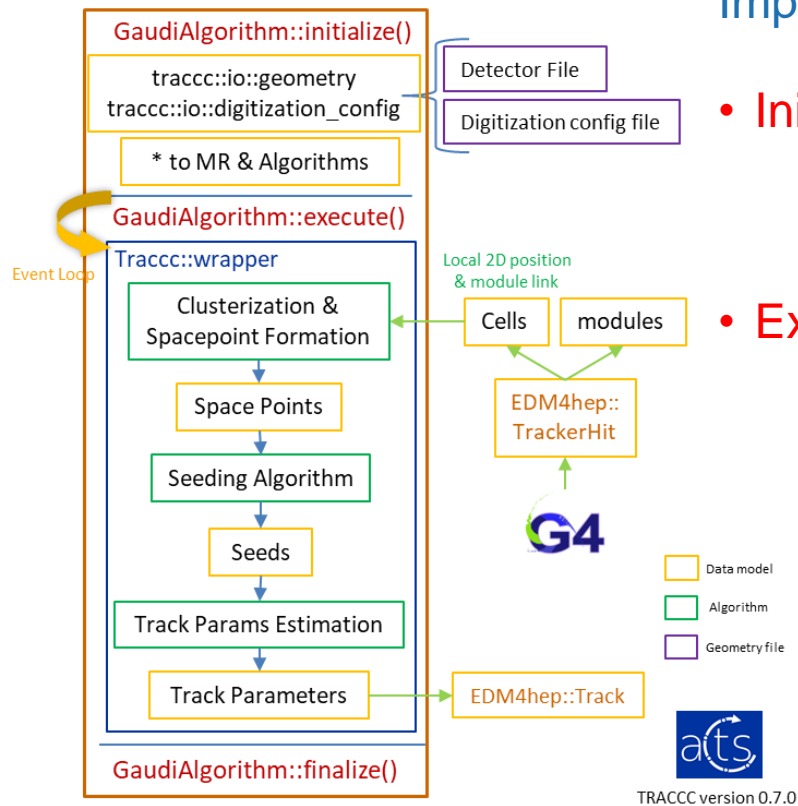
## Implement a Gaudi algorithm for seeding

## • Initialize():

- Read the **detector** file and **digitization** config file
- Initialize the **memory resource** and the **algorithms**

## • Execute():

- For each event, read hits and run the algorithms
  - EDM4hep::TrackerHit is converted to **Cells** & **modules**
  -  will only converted to **Cells** in TRACCC v0.16.0
- Since CEPCSW and TRACCC are using different compilers (Clang, GCC), respectively
  - Develop a **wrapper** for TRACCC SYCL algorithms
- Algorithms includes:
  - Clusterization & Spacepoint Formation (only CPU)
  - Seeding Algorithm
  - Track Params Estimation



Gaudi Algorithm using TRACCC reconstruction

# Outline

1

Introduction

2

Integration of TRACCC with CEPCSW

**3**

**Geometry & EDM**

4

Extension of seeding algorithm

5

Performance

6

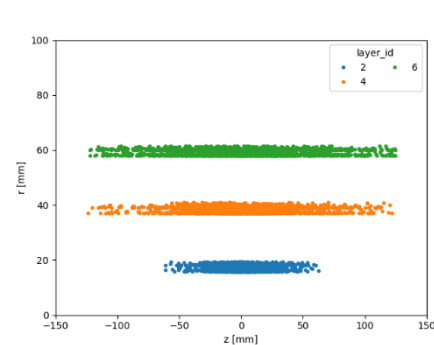
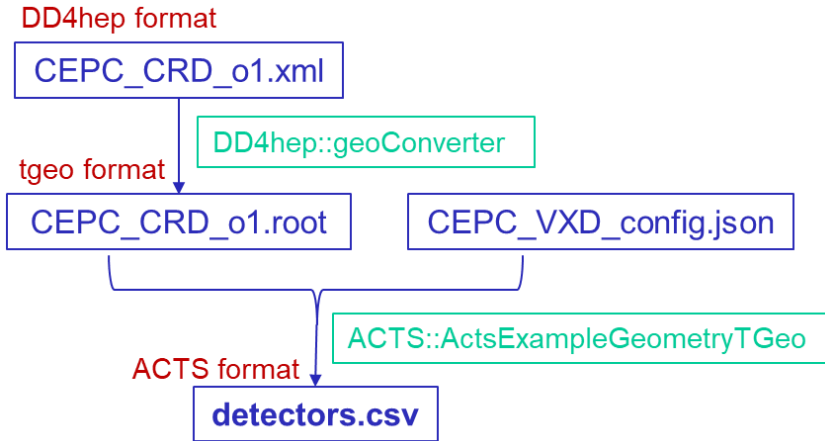
Summary

Geometry are prepared using various ACTS tools

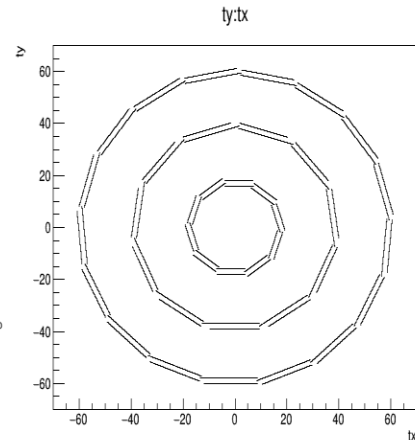
1. Convert the CEPC VTX geometry file (in DD4hep format) to TGeo format
2. The geometry file is translated into Acts::Surface objects using Acts::TGeoLayerBuilder, and is exported to a **detector file** by Acts::CsvTrackingGeometryWriter.
3. A **digitization config** file is written to provide the segmentation information of each surface.

### Verification:

Use Fast ATLAS Track Simulation (FATRAS) & ACTS' digitization tool to produce full simulation information and generate cells.



Gid of fatras of ACTS



X-Y projection of VTX

## Cell ID conversion between EDM4hep & TRACCC

- CEPCSW cell id needs to be converted into the TRACCC gid to retrieve correct geometry

### CEPCSW cell id:

Layer: {0,1,2,3,4,5} # Indicate 6 layers from inside to outside

Module: { L0: 0-9, L1: 0-9, L2: 0-10, L3: 0-10, L4: 0-16, L5: 0-16}

# Indicate ladders in the  $\phi$  direction

Sensor: 0

Barrelside: 1 for  $z > 0$  else -1

# one ladders has 2 sensors separated by  $z$

### TRACCC gid:

Volume: {3}

Boundary: 0

Layer: {2, 4, 6} # adjacent layers are treated as the same layers

Approach: 0

Sensitive: {L2: 1-40, L4: 1-44, L6:1-68}

The sensitive counts from  $z>0$  to  $z<0$ , then counts in  $\phi$  direction (the order is same to CEPC), and then counts from inner to outer layers.

```

uint64_t barrelsign = (n_barrelside - 1) / 1 + 2;
uint64_t actqvolnum = VXD_actq_volnum_id;
uint64_t acts_boundary = 0;
uint64_t acts_layer = 2 * (n_layer -> 1) + 2;
uint64_t acts_approach = 0;
uint64_t acts_sensitive = VXD_numOfSensors_per_layer[n_layer -> 1] * (n_layer & 1)
+ 2 * n_module + barrelsign;

//set acts geometry identifier
acts::GeometryIdentifier module0gid;
module0gid.setvolume(acts_volnum);
module0gid.setboundary(acts_boundary);
module0gid.setlayer(acts_layer);
module0gid.setapproach(acts_approach);
module0gid.setsensitive(acts_sensitive);
  
```

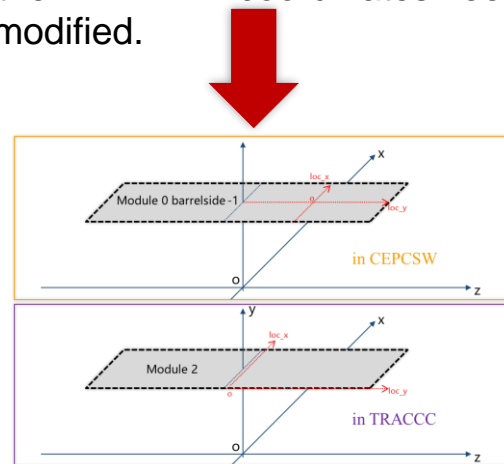
Converter for VXD gid

## Cells local position:

CEPCSW:

- take center point as the origin
- use the lower left corner as the origin.

So the cells' local coordinates need to be modified.

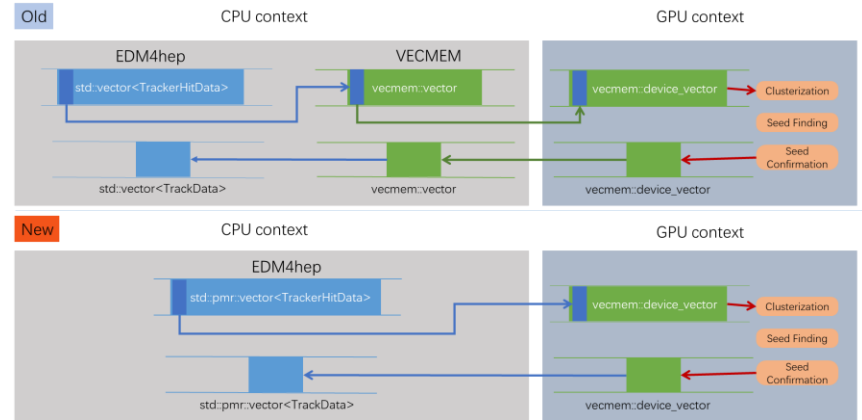


Adjust the local coordinates for the difference between CEPCSW & TRACCC

## Common memory for EDM4hep & VecMem

We want TRACCC to use hits data from EDM4hep **directly** !!

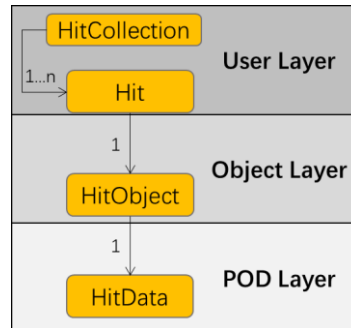
- TRACCC uses **VecMem** as the vectorised data model across multiple device types.
- EDM4hep and VecMem may use the same storage format (**std::pmr::vector**), so TRACCC can directly use the hit data with no data-copy.



Modified data transfer process

## Modify the data storage format of PODIO

EDM4hep is generated by PODIO, so we modify the DataContainer of PODIO.



Add **Collection** layer interfaces:

```
std::pmr::vector<{{ class.bare_type }}Data> data()
```

Add **CollectionData** layer interfaces:

```
{{ class.bare_type }}DataContainer getdata();
```

Modify the **DataContainer** storage format (vector → pmr::vector)

```
using {{ class.bare_type }}ObjPointerContainer = std::deque<{{ class.bare_type }}Obj*>;
using {{ class.bare_type }}DataContainer = std::pmr::vector<{{ class.bare_type }}Data>;
```

Layout of the PODIO storage format We add interfaces to get `pmr::vector` directly.

## Geometry & EDM

### Customized EDM4hep data collection

- Define a data collection whose member is completely the same as the EDM of TRACCC
- So we can directly use `edm4hep::ACTSCells` as the input of TRACCC.

### Verification

- Now TRACCC can directly read the simulated hits from Geant4 which is stored in EDM4hep format
- No non-essential data-copy is needed

```
#----- ACTSCells
edm4hep::ACTSCells:
  Description: "Cells for reconstruction in TRACCC Project"
  Author: "Yizhou Zhang, IHEP"
  Members:
    - uint32_t channel0 //channel0
    - uint32_t channel1 //channel1
    - float activation //activation
    - float time //time
    - uint32_t module_link //module_link
```

Modified edm4hep.yaml

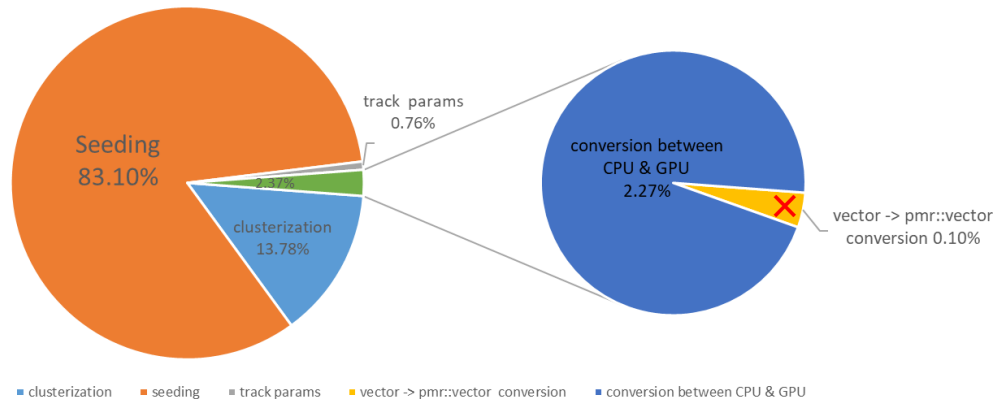
```
Running Seeding on device: Quadro RTX 8000
Initializing ...
EventLoopMgr WARNING Unable to locate service "EventSelector"
EventLoopMgr WARNING No events will be processed from external input.
ApplicationMgr INFO Application Manager Initialized successfully
ApplicationMgr INFO Application Manager Started successfully
TraccRun INFO begin execute TraccRun
TraccRun INFO reading hits from csv
TraccRun INFO the size of the csv's cells vector: 199547
TraccRun INFO creating edm4hep::ACTSCellsCollection
TraccRun INFO the address of the cells vector: 3963cf0
TraccRun INFO the size of the cells vector: 199547
TraccRun INFO running tracc
TraccRun INFO the address of the cells vector: 3963cf0
TraccRun INFO the size of the cells vector: 199547
Tracc Success
=>Elapsed times...
  Clusterization (sycl) 5 ms
  Seeding (sycl) 4 ms
  Track params (sycl) 0 ms
  Wall time 11 ms
TraccRun INFO event 0 success
```

In CEPCSW alg

In TRACCC alg

The address of `pmr::vector` does not changed.

TRACCC Seeding Time Cost (50tracks/event, 0.3% noise rate)



Reduce time overhead of vector  $\rightarrow$  pmr::vector conversion

# Outline

1

Introduction

2

Integration of TRACCC with CEPCSW

3

Geometry & EDM

4

Extension of seeding algorithm

5

Performance

6

Summary

## 4

## Extension of seeding algorithm

CEPC VTX detector:

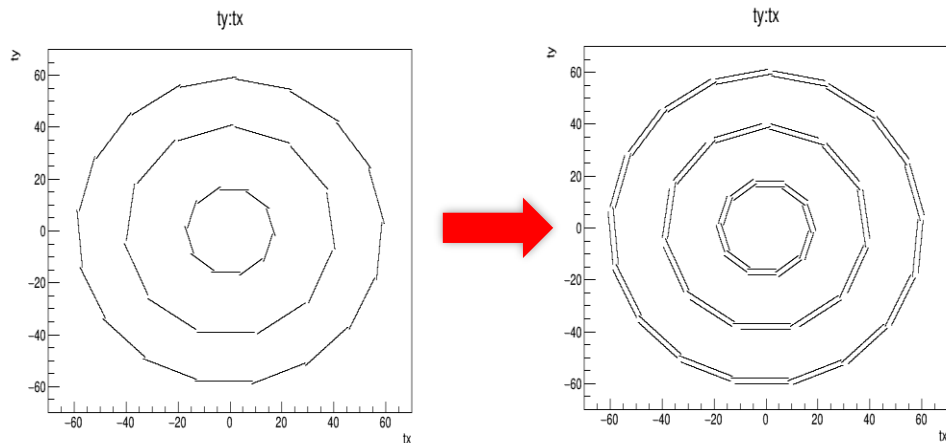
- Two sides of each layer have sensors
- A single seed contains 6 space points

Default TRACCC seeding alg:

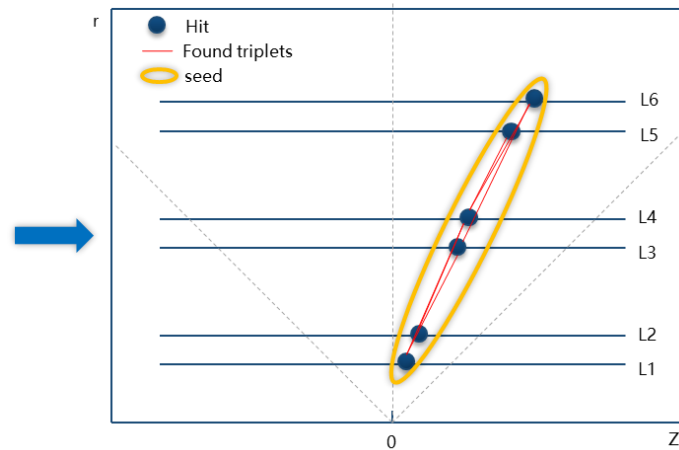
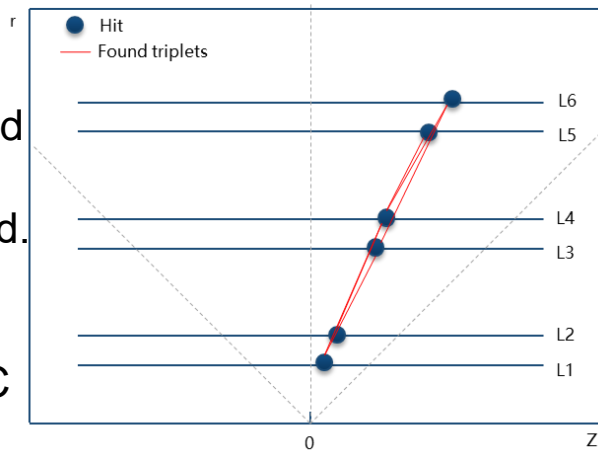
- creates 3-space-point seeds

→ The seeding alg needs to be extended for 6-space-points case.

## 6-layers seeds finding



## Seed Formation

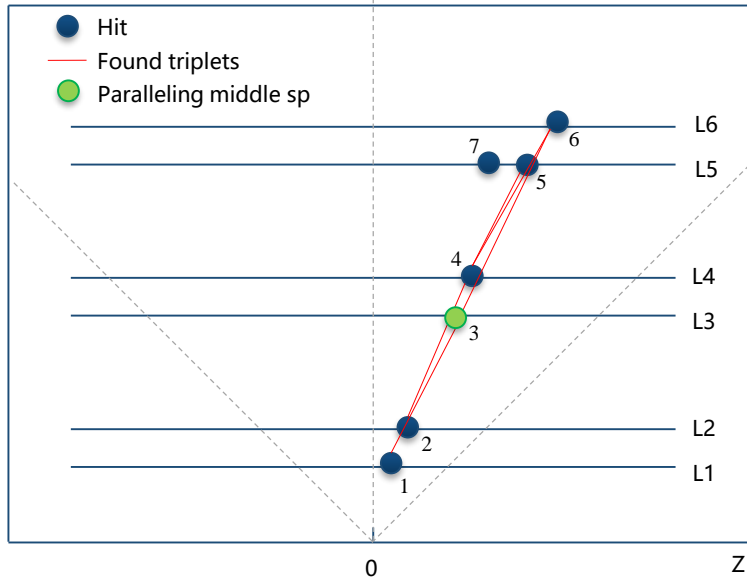


## Seed Formation Alg:

After seeding, combine the found triplets that sharing the same space-points into a “bigger” seed.

We have implemented **Seed Formation algorithm** in TRACCC





In GPU

Example:

Paralleling for hit 3

	bottom	middle	top
	1	3	5
lowest $d_0$	1	3	6
	1	3	7
	2	3	5
	2	3	6
	2	3	7



Bot_inner	Bot_outer	Mid_inner	Mid_outer	Top_inner	Top_outer
1	2	3	3	5	6

{1, 2}  
1.radius() < 2.radius()

{5, 6} | {7, 6}  
5.radius() < 6.radius()

## 6-layers seeds finding: Seed Formation steps in GPU

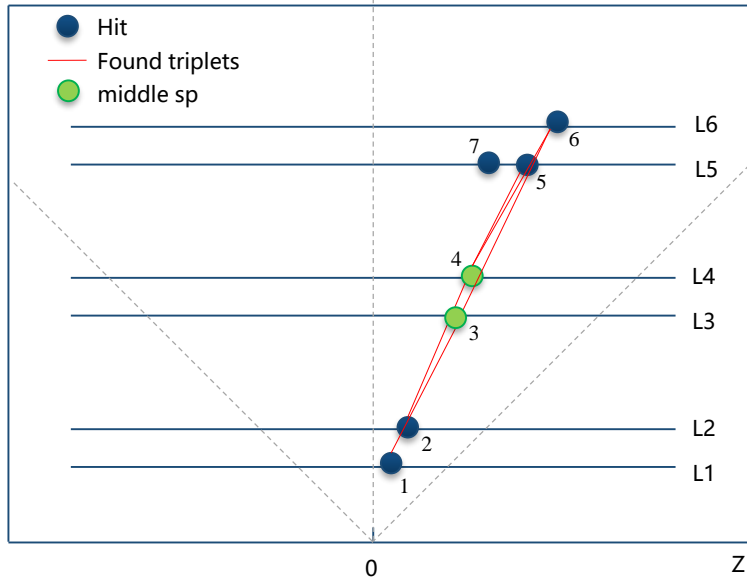
For each middle space point in parallel:

1. pick the triplet with lowest impact params ( $d_0$ ) among all triplets where the middle sp is located
2. find the bottom sp & top sp that are closest to the bottom sp & top sp of the current triplet
3. form a new seed of 5 points and sort them according to their radius

## 4

## Extension of seeding algorithm

In CPU

For hit 3 

Bot_inner	Bot_outer	Mid_inner	Mid_outer	Top_inner	Top_outer
1	2	3	3	5	6

For hit 4 

+

Bot_inner	Bot_outer	Mid_inner	Mid_outer	Top_inner	Top_outer
1	2	4	4	5	6



Bot_inner	Bot_outer	Mid_inner	Mid_outer	Top_inner	Top_outer
1	2	3	4	5	6

 $\{3, 4\}$  $3.\text{radius}() < 4.\text{radius}()$ 

## 6-layers seeds finding: Seed Formation step in CPU

Iterate through all new 5-point seeds:

- if two seeds have the same bottom sp & top sp, merge both into hexaplets (6-layer seeds)

# Outline

1

Introduction

2

Integration of TRACCC with CEPCSW

3

Geometry &amp; EDM

4

Extension of seeding algorithm

5

**Performance**

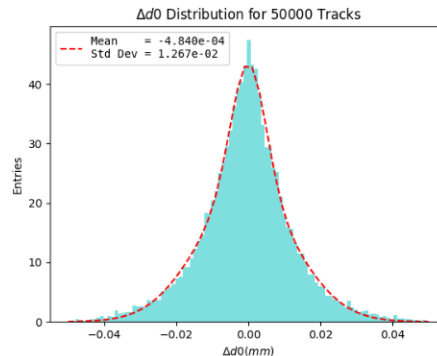
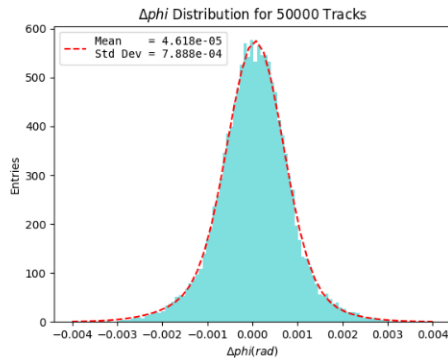
6

Summary

Particle: mu-/pi-/e-/K-/proton Energy: 10 GeV

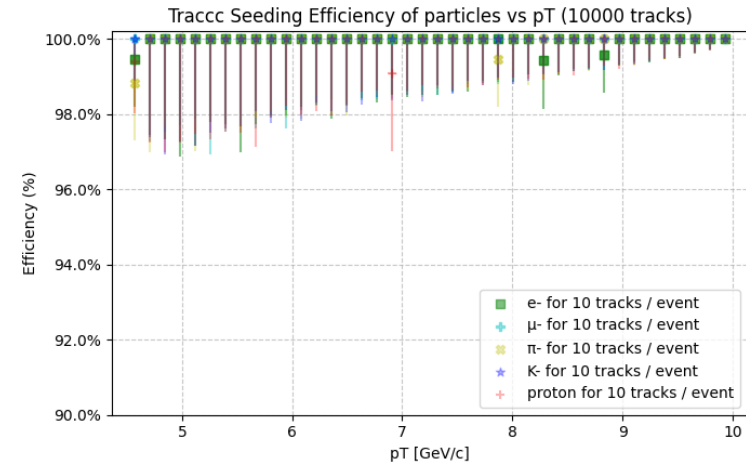
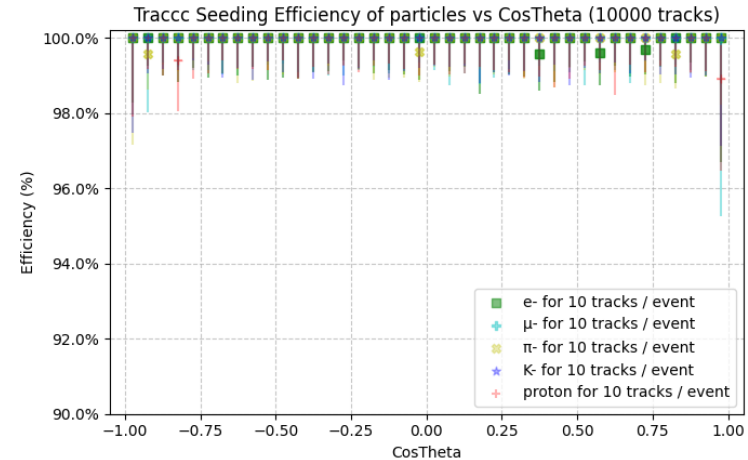
Number of events: 1000 Number of tracks per event: 10

- **Good seed**: half space-points of the seed are from the same particle.
- Pick tracks with polar angle  $|\cos\theta| < 0.921$  to avoid boundary effects.
- The seeding efficiency is above 99.5% without background for all types of particles
- Resolution of  $d_0/\phi$  is as expected



Difference between rec and sim track param

Track parameters include  $d_0$ ,  $\phi$  (particle: mu-)

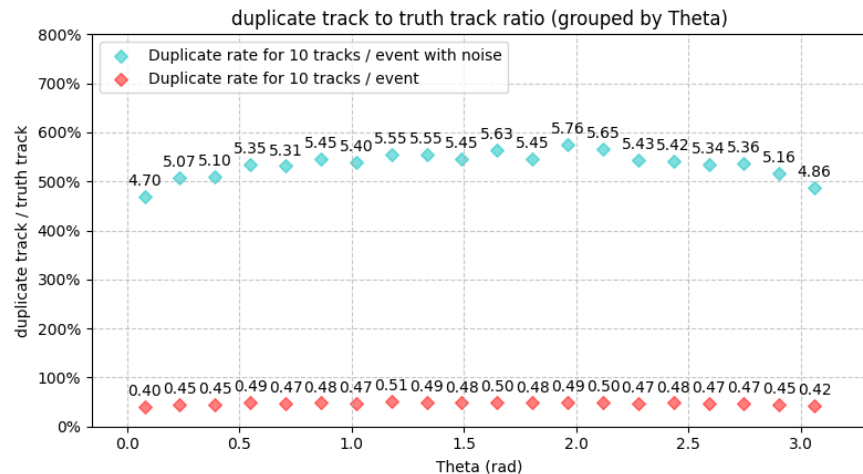
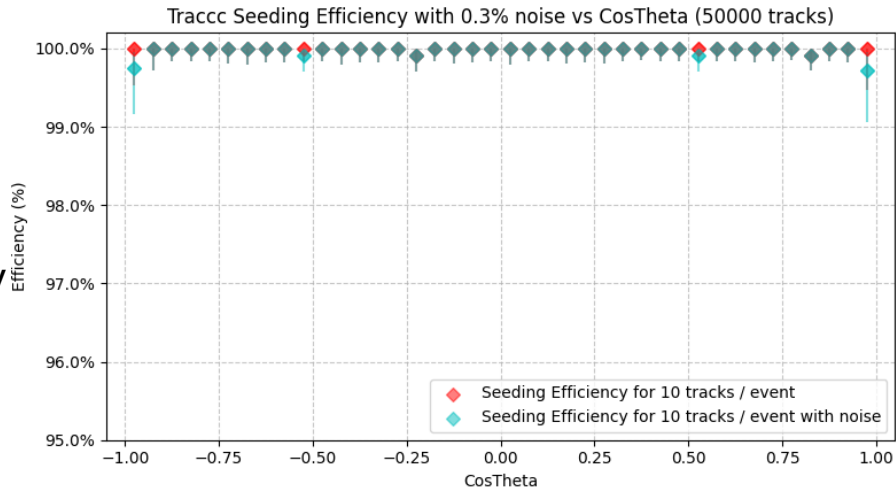


Particle: mu- Energy: 10 GeV

Number of events: 5000

Number of tracks per event: 10

- After adding 0.3% noise, reconstruction efficiency drops slightly.
  - Reason for the decrease in efficiency: If the noise and hit are too close, they may be grouped in the same space-point during clustering, which may result in \*wrong position\* or \*wrong particle id\*.
- Higher repetition rate after adding noise.
  - Why no-noise case has  $\approx 30\%$  repetition rate: When processing the Seed Formation algorithm, if there are two triplets from the same track that do not share points at all, a duplicate seed is generated.



## 5

## Performance

## Seeding efficiency versus # of track / event

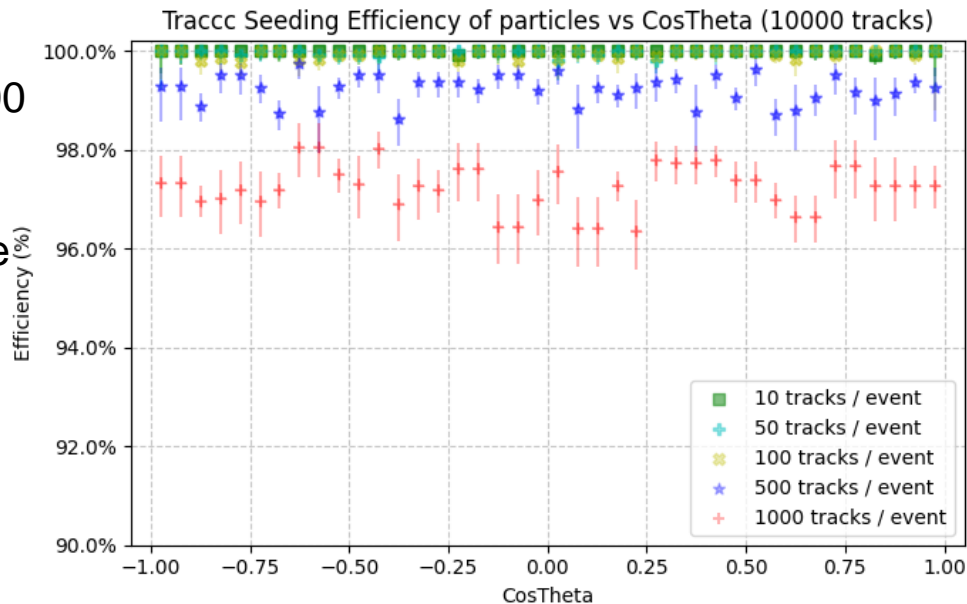
Particle: mu- Energy: 10 GeV

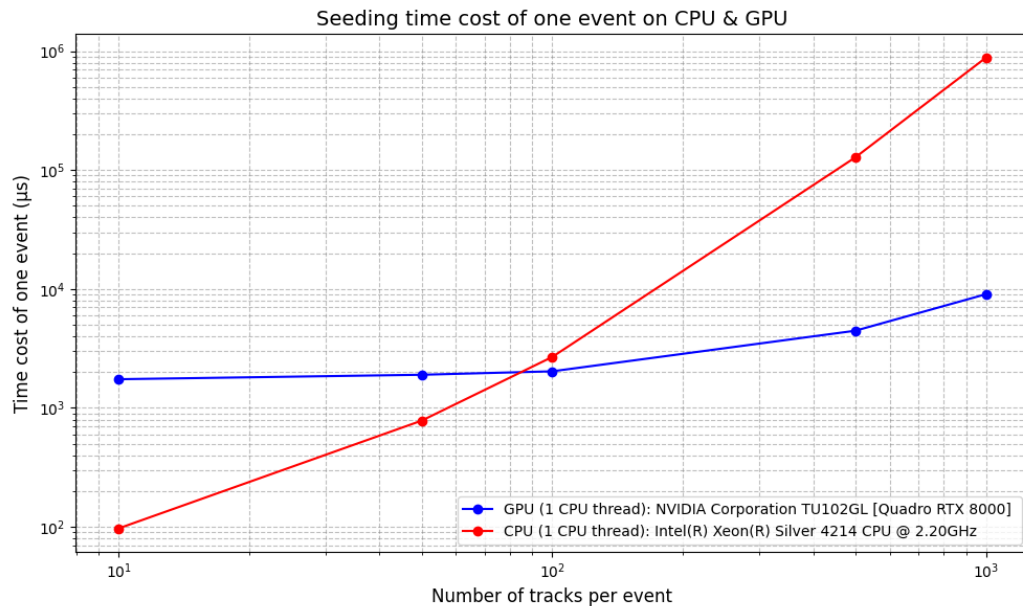
Total track num: 50000

Number of tracks per event: 10/50/100/500/1000

Conclusion:

- The **Seed Formation** algorithm may combine two triplets with different particle id into a Seed. And the marked particle id of that is defined as the mode of the particle id of all **space-points**, which causes the absence of the proper particle id.





## Computing evaluation of TRACCC seeding

Particle: mu- Energy: 10 GeV

Run TRACCC in heterogeneous device:

- CPU: Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz
- GPU: NVIDIA Corporation TU102GL [Quadro RTX 8000]

## We tested the computing efficiency on CPU&GPU with only single CPU thread

- Even in this circumstances, we can beat a single CPU with a single “workstation” GPU at 100 tracks’ event.
- With multiple CPU cores in use, GPU can only “win” at large pipe-up case.

# Outline

- 1 Introduction
- 2 Integration of TRACCC with CEPCSW
- 3 Geometry & EDM
- 4 Extension of seeding algorithm
- 5 Performance
- 6 Summary**



1. TRACCC has been applied to the CEPCSW for the **first time**.  
**If you want to apply TRACCC on your experiments:**
  - The geometry conversion can be easily done by ACTS Tools.
  - The EDM conversion needs **careful manually search** for one-to-one correspondence
  - The TRACCC's algorithm can be extended for the special detector
2. For the performance of TRACCC
  - The **physical performance** of the seeding algorithm is promising
  - GPU shows better **computing performance** than the CPU for large pile-up events

Previous Reports:

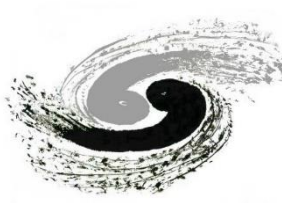
ACTS Developers Meeting:

<https://indico.cern.ch/event/1388561/>

<https://indico.cern.ch/event/1406633/> (Apply ACTS in the inner tracker of CEPC)

Acts Parallelization Meeting (TRACCC):

<https://indico.cern.ch/event/1356203/>



# Thank You

YiZhou Zhang, WeiDong Li, Xiaocong Ai, Tao Lin  
[zhangyz@ihep.ac.cn](mailto:zhangyz@ihep.ac.cn)  
The Institute of High Energy Physics, IHEP, China

CHEP2024  
Krakow, Poland  
23<sup>rd</sup> Oct 2024

Backup

## 2 Integration of TRACCC with CEPCSW

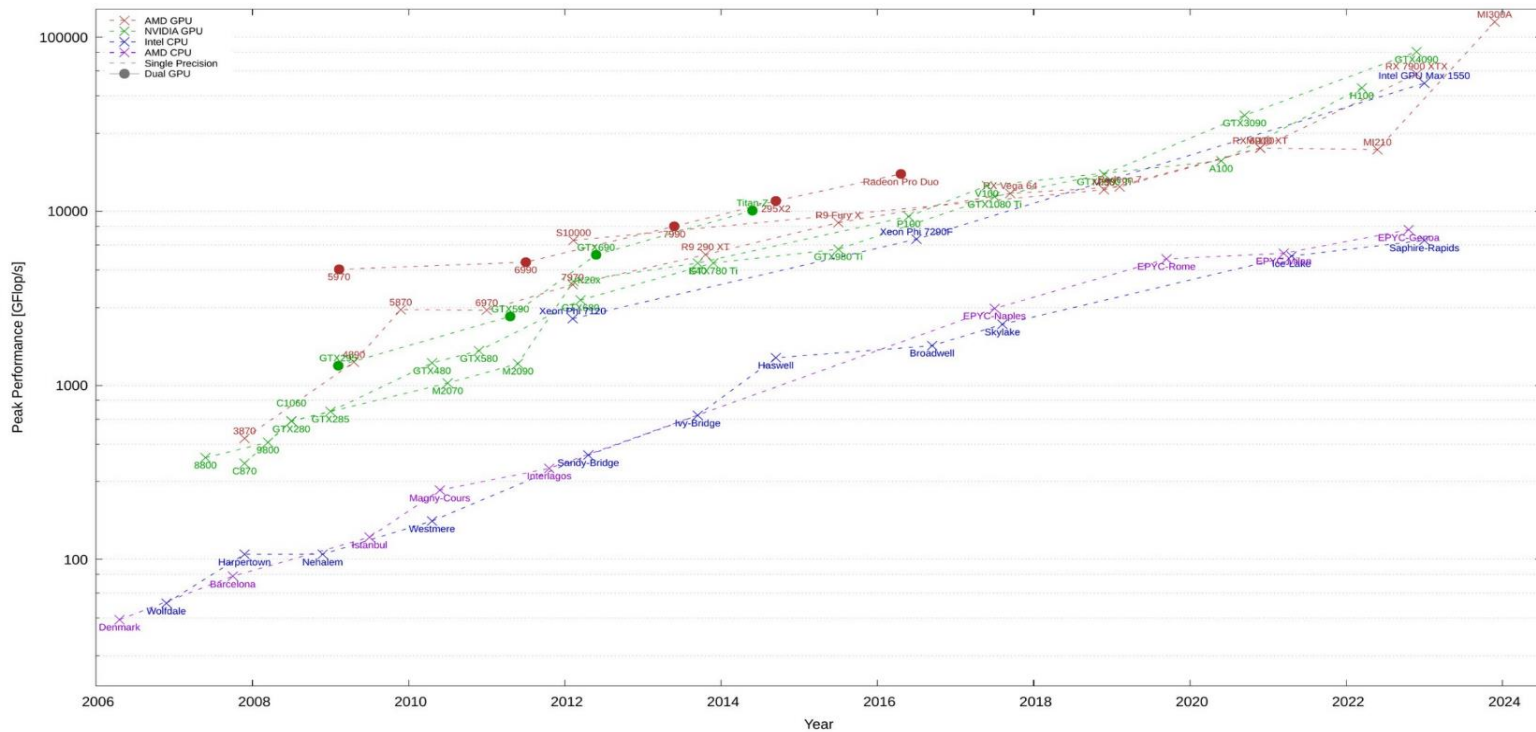
- Physical performance improvements brought by the algorithm
- 重复率下降
- 6个点的角度重建误差比3个点好

## 1

## Introduction

1. The performance of CPUs is still growing exponentially, although a bit slower than a few years ago.
  2. GPUs are growing at the same rate, with the offset remaining almost unchanged.
  3. Leveraging modern hardware and parallelization remains our main direction for algorithm development
- \* **TRACCC** is one of the ACTS R&D projects, which is developing full track reconstruction algorithms that can run on accelerators.

使用成本



# Geometry & EDM

## Data model conversion between EDM4hep & TRACCC

- CEPCSW cellid needs to be converted into the TRACCC cellid to retrieve the correct module.

### CEPCSW cellid:

Layer: {0,1,2,3,4,5} # Indicate 6 layers from inside to outside

Module: { L0: 0-9, L1: 0-9, L2: 0-10, L3: 0-10, L4: 0-16, L5: 0-16}

# Indicate ladders in the  $\phi$  direction

Sensor: 0

Barrelside: 1 for  $z > 0$  else -1

# one ladder has 2 sensors separated by  $z$

### TRACCC gid:

Volume: {3}

Boundary: 0

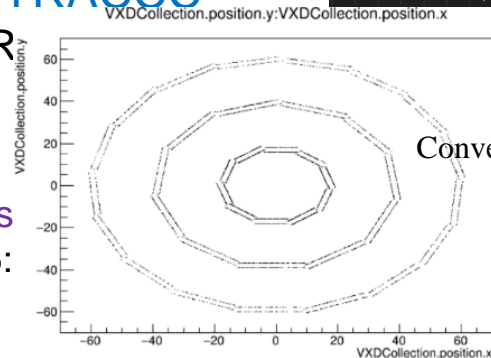
Layer: {2, 4, 6} # adjacent layers are treated as the same layers

Approach: 0

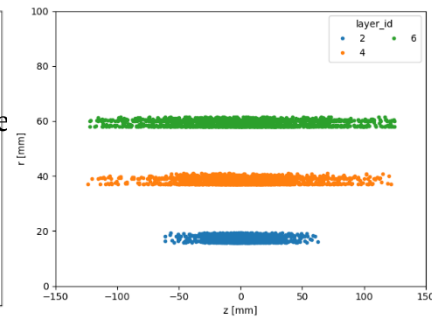
Sensitive: {L2: 1-40, L4: 1-44, L6:1-68}

The sensitive counts from  $z > 0$  to  $z < 0$ , then counts in  $\phi$  direction (the order is same to CEPC), and then counts from inner to outer layers.

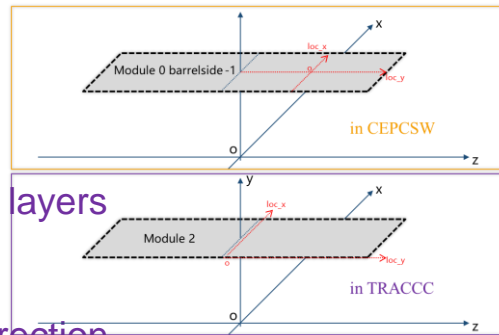
```
uint64_t barrel_sign = (m_barrelside == 1) ? 1 : 2;
uint64_t acts_volume = VXD_acts_volume_id;
uint64_t acts_boundary = 0;
uint64_t acts_layer = 2 * (m_layer >> 1) + 2;
uint64_t acts_approach = 0;
uint64_t acts_sensitive = VXD_NumOfSensors_per_layer[m_layer >> 1] * (m_layer + 1)
+ 2 * m_module + barrel_sign;
```



Generated hits of CEPC VXD by Geant4



Gid of fatras of ACTS



Adjust the local coordinates for the difference between EDM4hep & TRACCC

### Cells local position:

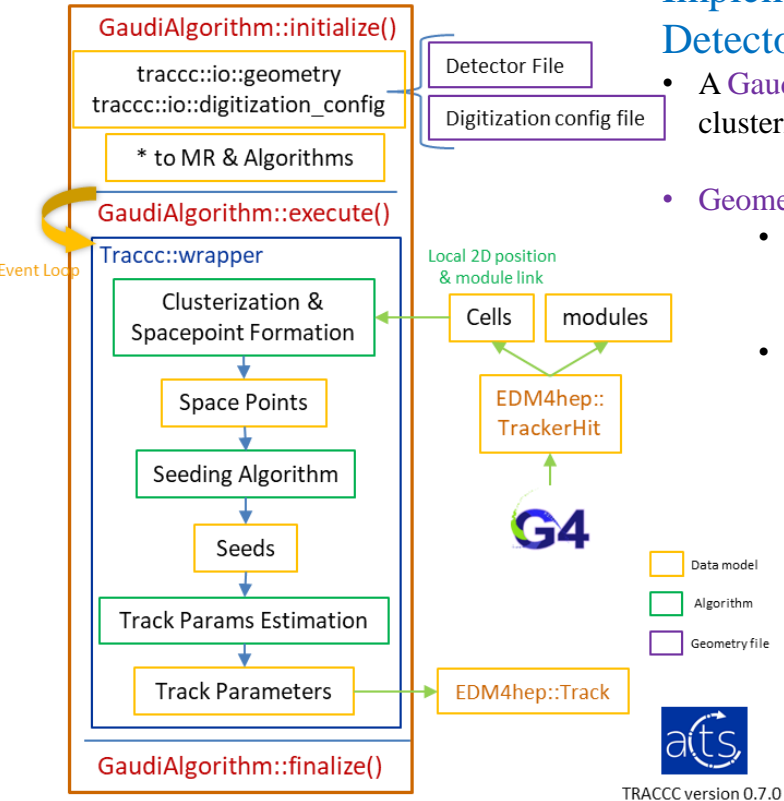
Geant4 take the center point as the origin, while TRACCC use lower left corner as the origin. So the cells' local coordinates need to be modified.

## Implementation of Seeding Algorithm with TRACCC for VerTex Detector (VTX) of the CEPC experiment

- A **Gaudi algorithm** has been developed for Seeding Algorithm, incorporating TRACCC clusterization, spacepoint formation, seed finding, and track params estimation algorithms.
- **Geometry** required for Seeding Algorithm are prepared using various ACTS tools
  - The CEPC geometry file, originally in DD4hep format, is first converted to TGeo format. It is then translated into Acts::Surface objects using Acts::TGeoLayerBuilder, and is exported to a **detector file** by Acts::CsvTrackingGeometryWriter.
  - A **digitization config file** is written to provide the segmentation information of each surface.

### Gaudi Algorithm workflow

- **Initialize():** Build the **silicon detector description** by detector file and digitization config file, and initialize the memory resource and TRACCC's algorithms.
- **Execute():** Read hits and run seeding algorithm for each events
  - An **wrapper** of TRACCC SYCL algorithms is developed since CEPCSW & TRACCC using different compilers (Clang & GCC).
  - In CEPCSW, hits are generated by Geant4 and stored in EDM4hep format. Therefore, **data model conversion** between EDM4hep & TRACCC is required before and after the seeding.



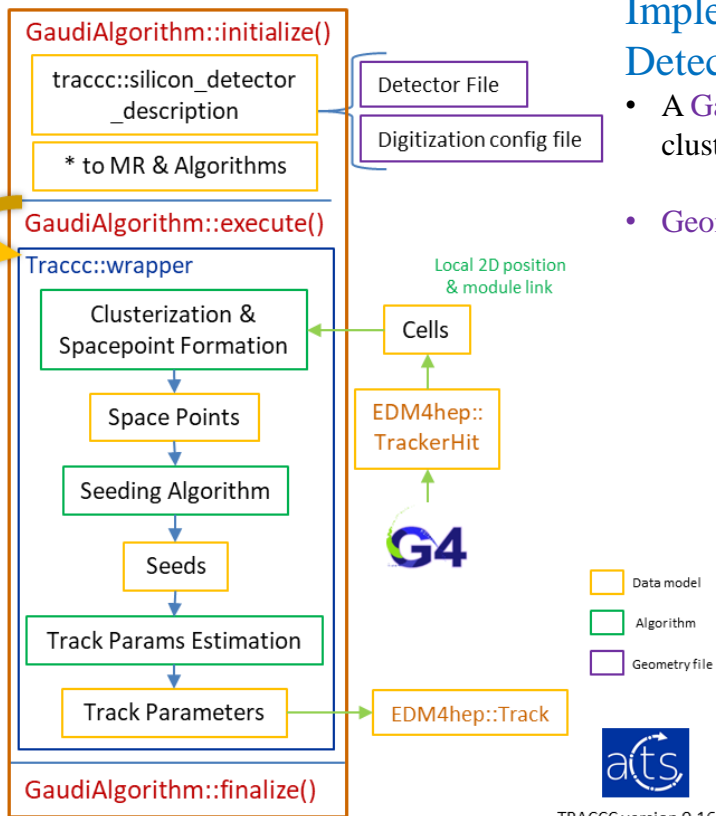
Gaudi Algorithm using TRACCC reconstruction

## Implementation of Seeding Algorithm with TRACCC for VerTex Detector (VTX) of the CEPC experiment

- A **Gaudi algorithm** has been developed for Seeding Algorithm, incorporating TRACCC clusterization, spacepoint formation, seed finding, and track params estimation algorithms.
- **Geometry** required for Seeding Algorithm are prepared using various ACTS tools
  - The CEPC geometry file, originally in DD4hep format, is first converted to TGeo format. It is then translated into Acts::Surface objects using Acts::TGeoLayerBuilder, and is exported to a **detector file** by Acts::CsvTrackingGeometryWriter.
  - A digitization config file is written to provide the segmentation information of each surface.

### Seeding workflow

- Data model conversion between EDM4hep & TRACCC is required before and after Seeding Algorithm (page 9)
  - CEPCSW cellid needs to be converted into the Acts gid to retrieve the correct module.
- using one common memory for both EDM4hep and VecMem to avoid the overhead from data copy (page 10)
- Using one common memory for both EDM4hep and VecMem to avoid the overhead from data copy (page 10)



TRACCC version 0.16.0

Gaudi Algorithm using TRACCC reconstruction