# INTT software

Takashi Hachiya

Nara Women’s University

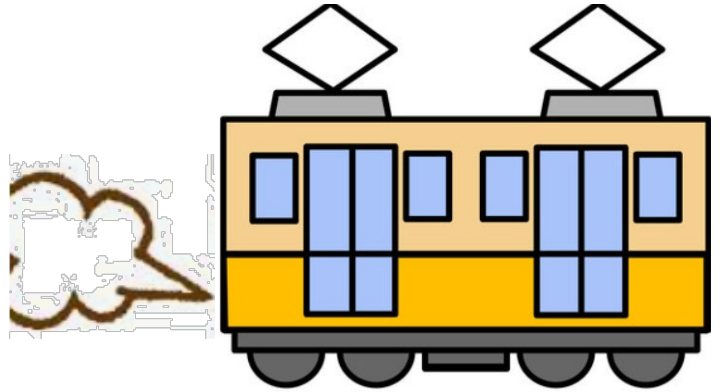
# Before Introduction

Hello, Takashi!
Can we discuss about INTT workshop session a little bit?

午後 4:04

既読

Sure! I



One thing we would like to get is, review session for **INTT related software**.

The point of this session is, we don't have to learn how to use during the session. We want to get some info, **what kind of software(analysis module) we have in sPHENIX github**, and what is purpose fhat module.

Oh, can we have our InttMapping review as well?

2024/11/25

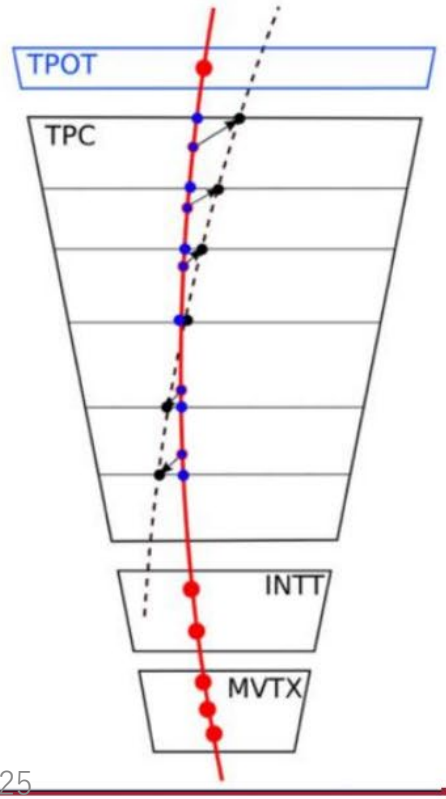
# Purposes of INTT

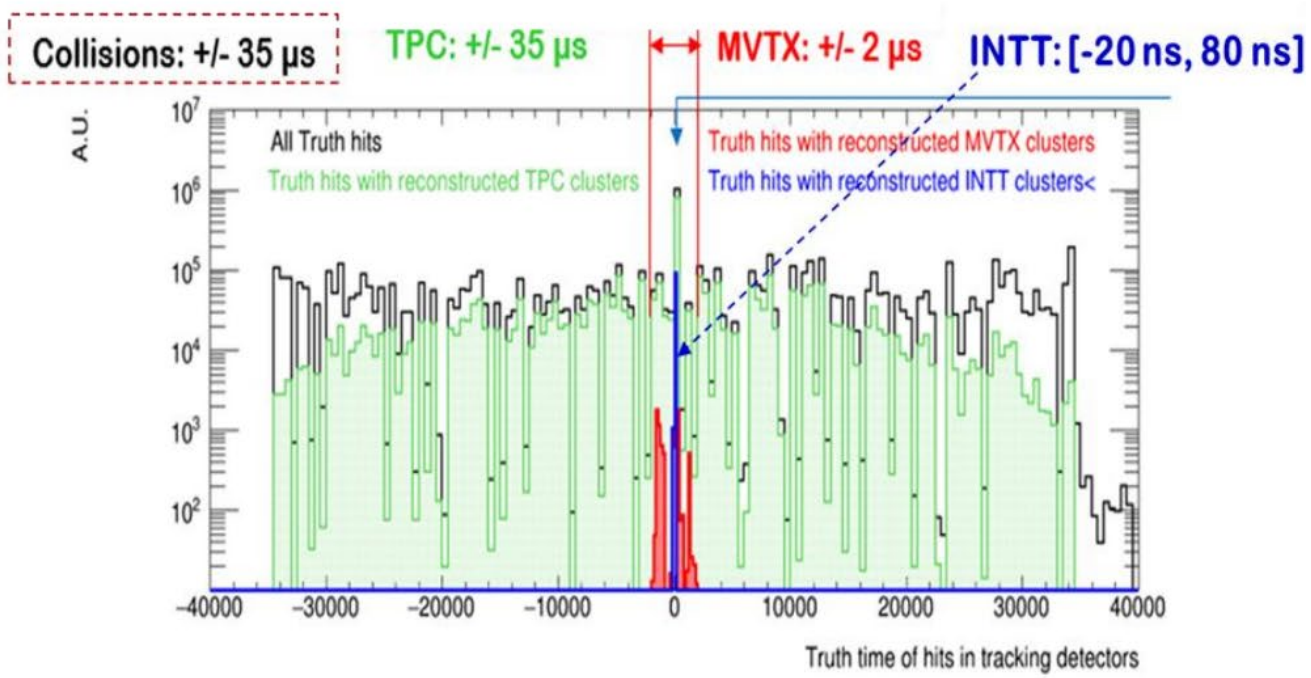
- Charged particle tracking
  - Together with MVTX, TPC, TPOT

Beam crossing Identification
Treso ~ 100ns (INTT), ~5us(MVTX), ~80us (TPC)

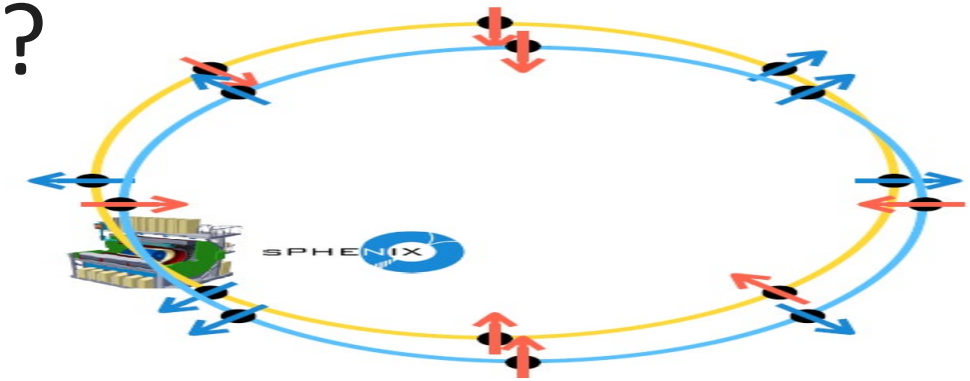
- More?  Z-vertex, dN/deta,  Reaction plane, Centrality for streaming data

## Tracking Points and Timing



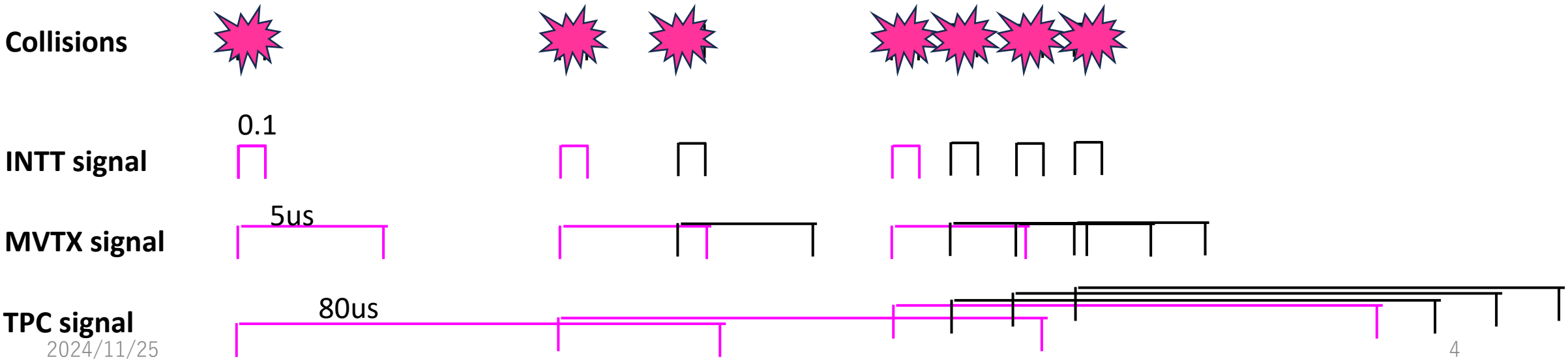
# Why is beam crossing important?



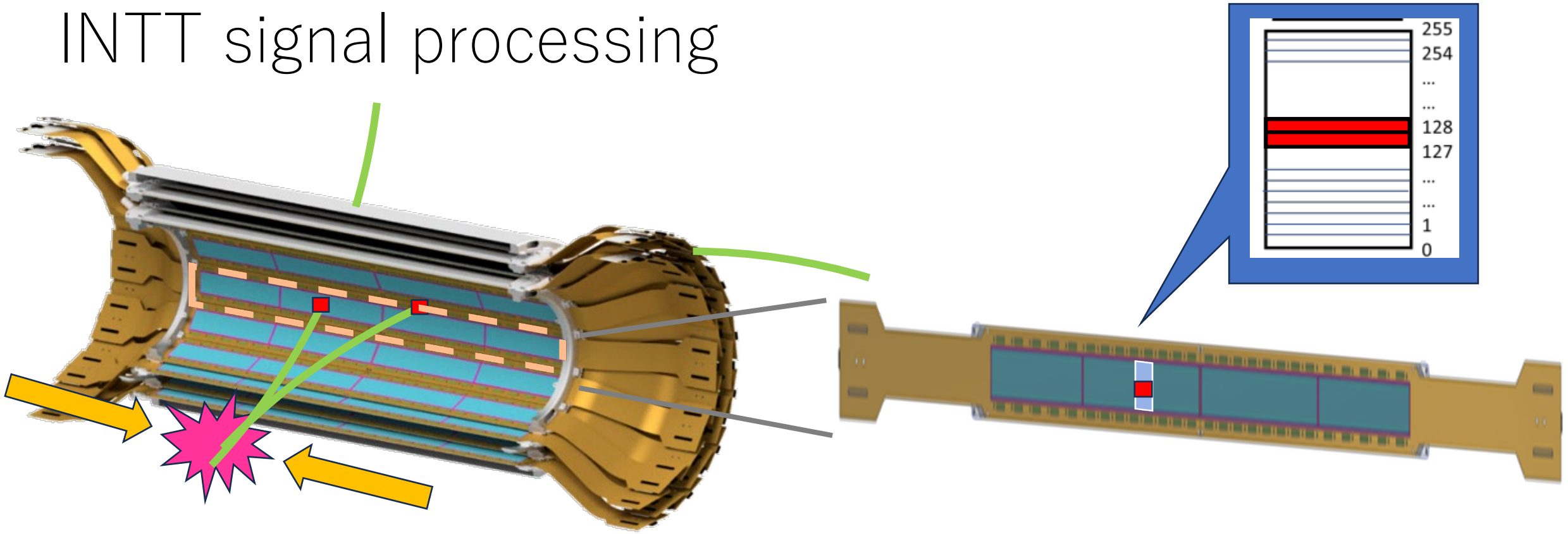

- The beam has a bunch structure
  - Bunch length is 106ns
  - Nbunchs is 120

- Treso ~ 0.1us(100ns) (INTT), ~5us(MVTX), ~80us (TPC)
  - Signals from different beam crossing mixed up
  - Physics we aim to measure is a quantity per COLLISION

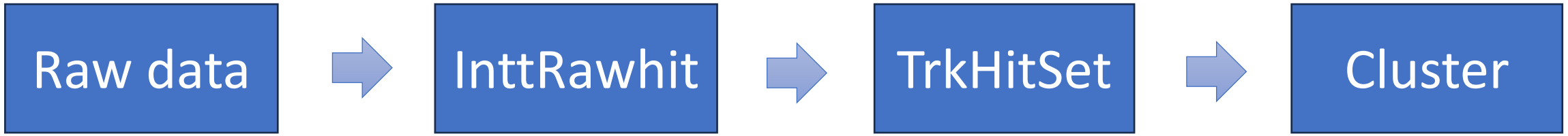
Timing Diagram



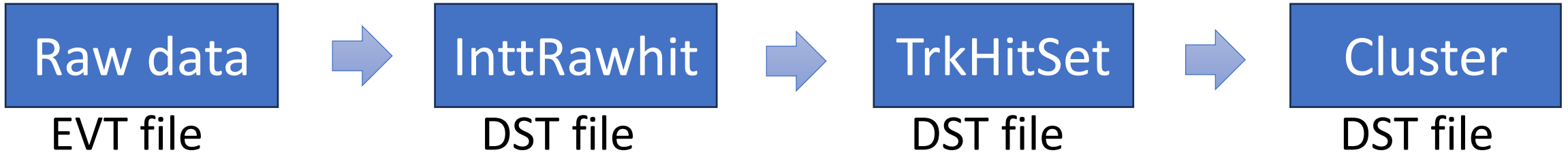
# INTT signal processing




- Raw data is converted to 3D hit position step by step

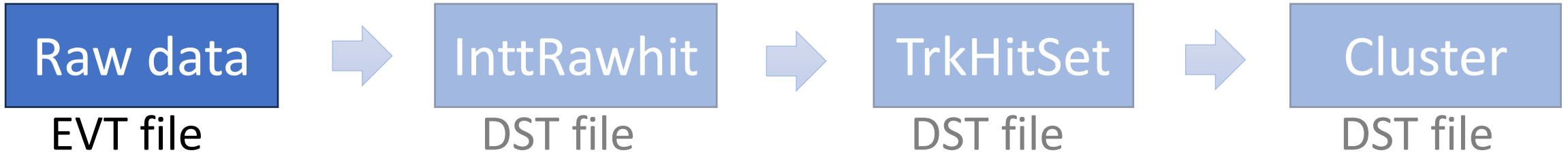


# INTT data (format)




- Raw data:  Binary data from INTT detector (FELIX)
  - a 32bit binary contains hit channel, chip, ladder, FELIX, ADC are stored in
  - Each FELIX server produce own EVT file ->  INTT has 8 evt files
- InttRawhit: DST style data (C++ class)
  - Hit channel, chip, ladder, FELIX, adc are stored in the variables
  - Channel, chip, ladder, FELIX ID are defined by INTT team
- TrkrHitSet: DST style data but sPHENIX format for global tracking
  - Similar w/ InttRawhit
  - Channel, chip, ladder, FELIX ID are defined by sPHENIX tracking
- Cluster : DST style data
  - adjacent hits grouped to single cluster

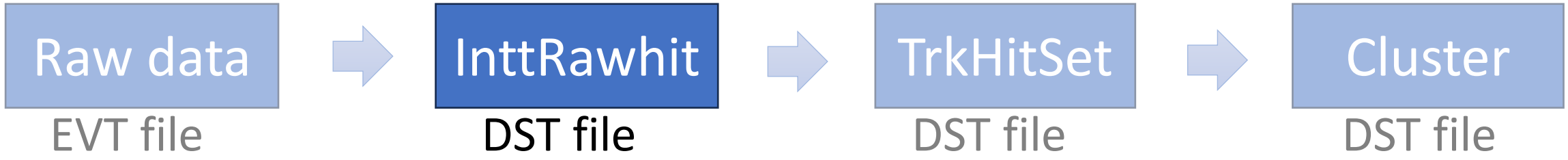
# INTT data (format)




- Raw data:  Binary data from INTT detector (FELIX)
  - a 32bit binary contains hit channel, chip, ladder, FELIX, ADC are stored in
  - Each FELIX server produce own EVT file ->  INTT has 8 evt files

| | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | LSB16, MSB16が入れ替わっている |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | Header | | | | | | | | fee | | | | len | | | | Header | | | | | | | | | | | | | | | | header=0xF000CAF0 |
| 2 | 1 | Hit-Header | | | | | | | | BCO | | | | | | | | Hit-Header | | | | | | | | | | | | | | | | Hit-header |
| 3 | 2 | BCO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | 3 | adc | | | | chip_id | | | | chan_id | | | | | | | | | | | | AMPL | | | full ROC | full FPX | | FPHX_BCO | | | | | | hit |
| 5 | 4 | adc | | | | chip_id | | | | chan_id | | | | | | | | | | | | AMPL | | | full ROC | full FPX | | FPHX_BCO | | | | | | hit |
| 6 | 5 | adc | | | | chip_id | | | | chan_id | | | | | | | | | | | | AMPL | | | full ROC | full FPX | | FPHX_BCO | | | | | | hit |
| 7 | 6 | adc | | | | chip_id | | | | chan_id | | | | | | | | | | | | AMPL | | | full ROC | full FPX | | FPHX_BCO | | | | | | hit |
| 8 | 7 | adc | | | | chip_id | | | | chan_id | | | | | | | | | | | | AMPL | | | full ROC | full FPX | | FPHX_BCO | | | | | | hit |

# INTT raw data (format)



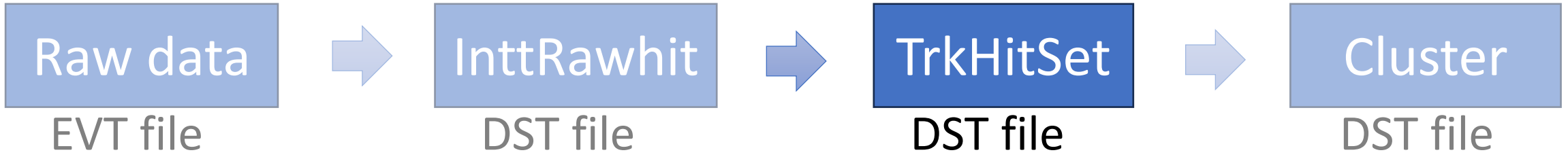

- InttRawhit: DST style data (C++ class)
  - Hit channel, chip, ladder, FELIX, adc are stored in the variables
  - Channel, chip, ladder, FELIX ID are defined by INTT team

InttRawhit in github

```
protected:
```

| | |
|---|---|
| BCOFULL | `uint64_t bco = std::numeric_limits<uint64_t>::max();` |
| FELIX_ID | `int32_t packetid = std::numeric_limits<int32_t>::max();` |
| | `uint32_t word = std::numeric_limits<uint32_t>::max();` |
| Ladder | `uint16_t fee = std::numeric_limits<uint16_t>::max();` |
| Channel | `uint16_t channel_id = std::numeric_limits<uint16_t>::max();` |
| Chip | `uint16_t chip_id = std::numeric_limits<uint16_t>::max();` |
| Adc | `uint16_t adc = std::numeric_limits<uint16_t>::max();` |
| BCO | `uint16_t FPHX_BCO = std::numeric_limits<uint16_t>::max();` |
| | `uint16_t full_FPHX = std::numeric_limits<uint16_t>::max();` |
| | `uint16_t full_ROC = std::numeric_limits<uint16_t>::max();` |
| AMPL for calib | `uint16_t amplitude = std::numeric_limits<uint16_t>::max();` |
| Added in v2 | `uint32_t event_counter = std::numeric_limits<uint32_t>::max();` |

# INTT TrkrHitSet(format)




- TrkrHitSet: DST style data but sPHENIX format for global tracking
  - Similar w/ InttRawhit
  - Channel, chip, ladder, FELIX ID are defined by sPHENIX tracking

TrkrDef.h, InttDef.h, TrkrHitv2.h

TrkrHitSet : double array [hitsetkey][hitkey] = ADC

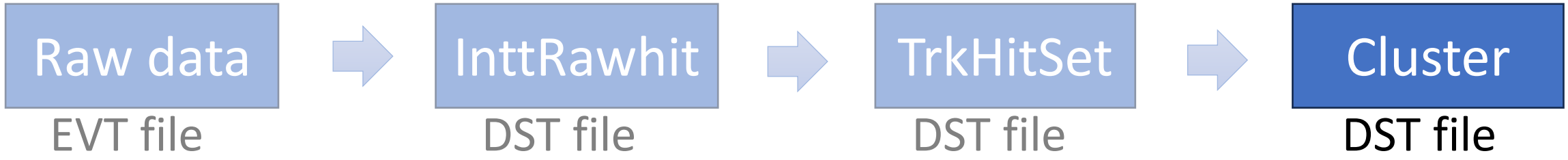
hitsetkey :

| 31-24 | 23-16 | 15-14 | 13-10 | 9-0 |
|---|---|---|---|---|
| TrackerID | Layer | Z-ID | Ladder phi | BCO (time bucket) |

hitkey : chip[] +channel[0-255 7:0]+BCO[]

Tracker ID and Layer are commonly used for INTT, MVTX, TPC, TPOT

# INTT Cluster(format)




TrkrCluster : array [cluskey] of cluster

cluskey : hitsetkey[63:32] + clusid[31:0]

hitsetkey :

| 31-24 | 23-16 | 15-14 | 13-10 | 9-0 |
|---|---|---|---|---|
| Tracker ID | Layer | Z-ID | Ladder phi | BCO (time bucket) |

Clusid: i_cluster in the sensor

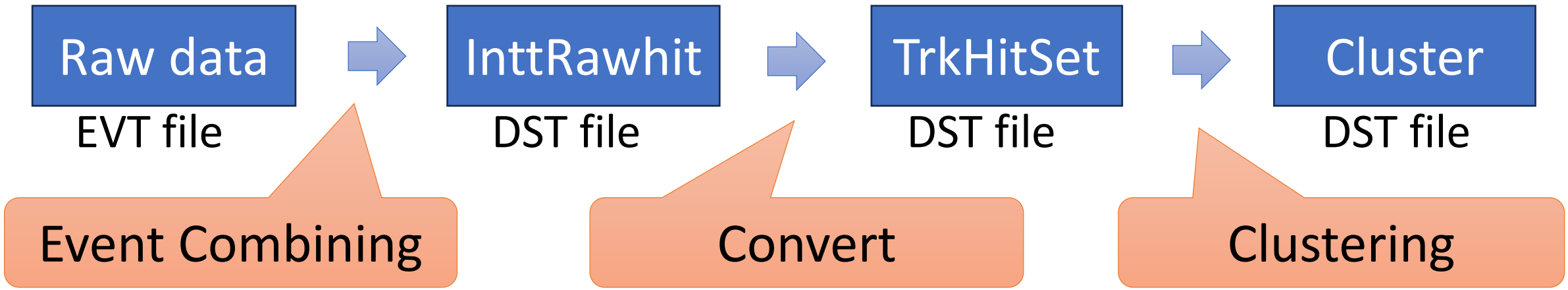
TrkrDef.h in github

TrkrClusterv5 in github

Cluster variables

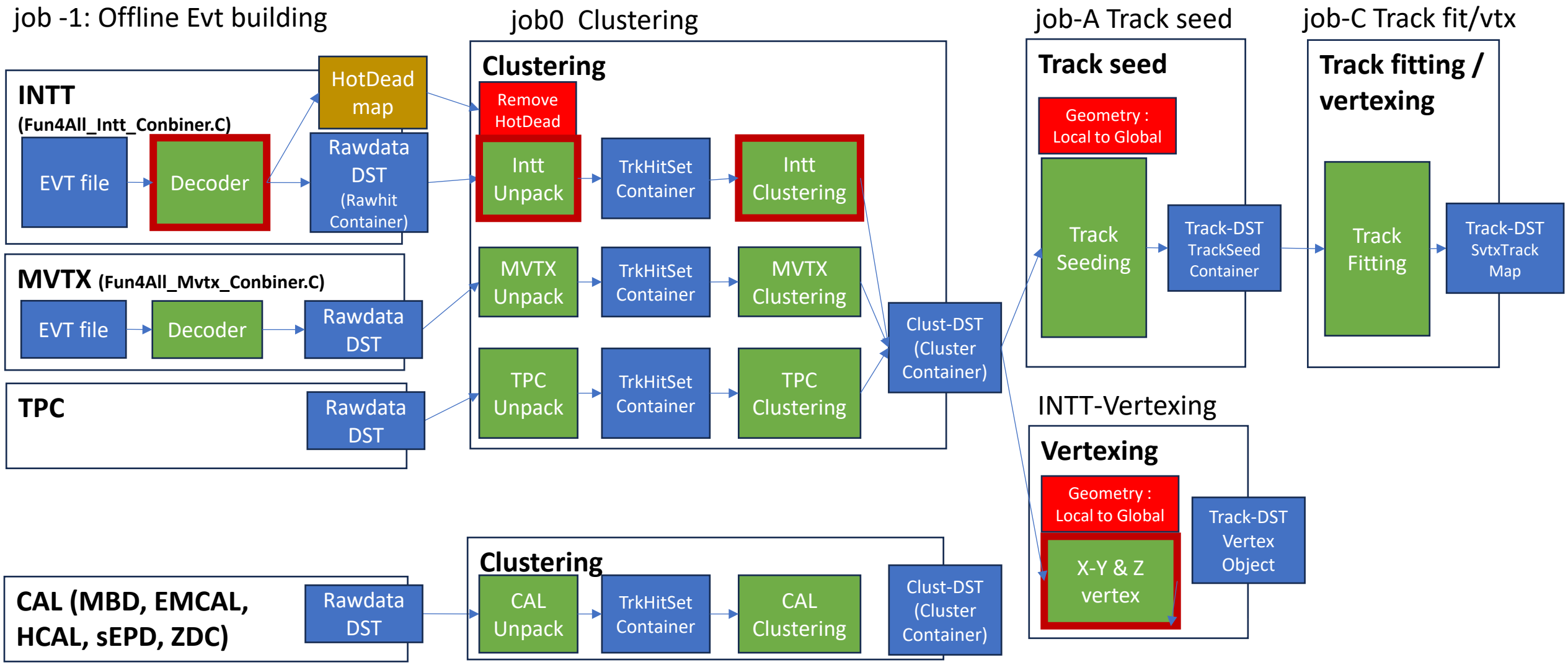
```
protected:
 float m_local[2]{};                    //< 2D local
 TrkrDefs::subsurfkey m_subsurfkey;     //< unique i
 float m_phierr;
 float m_zerr;
 unsigned short int m_adc;              //< cluster sum ad
 unsigned short int m_maxadc;           //< cluster sum ad
 char m_phisize;                        // 8bit
 char m_zsize;                          // 8bit
 char m_overlap;                        // 8bit
 char m_edge;                           // 8bit - cumul 2*
```

# INTT data (format)




- Event Combining: SingleInttPoolInput.h
  - Convert rawdata to InttRawhit
  - Synchronize 8 FELIXs w/ BCOFULL
  - Synchronize INTT with GL1 and other detectors w/ BCOFULL
- Convert
  - Convert InttRawhit to TrkrHitSet
  - Hot/dead, BCO cut calibration applied to remove “BAD” hits
  - BCO -> Time (Timebacket)
- Clustering
  - Clustering hits sensor by sensor and time by time

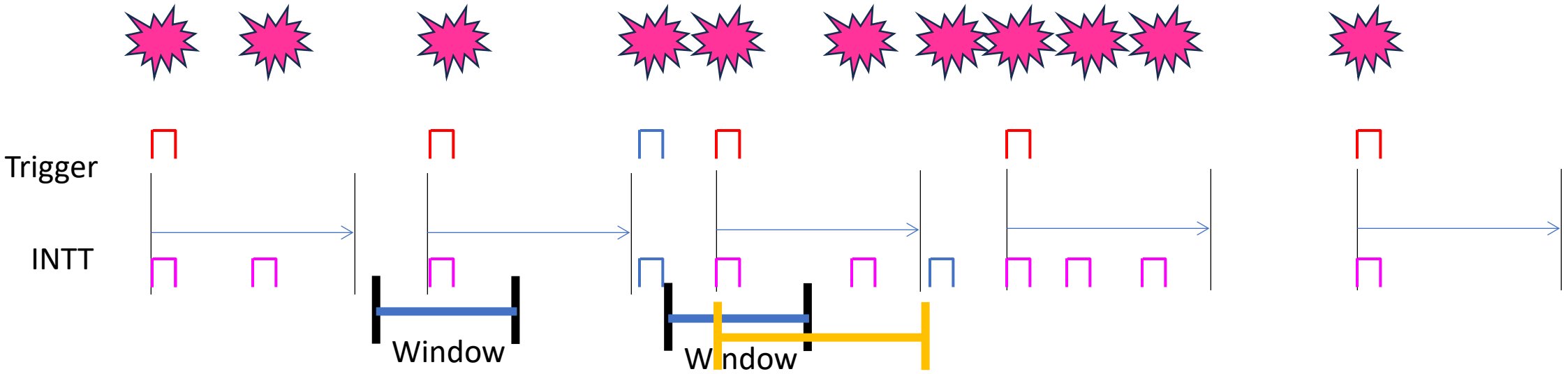
# Workflow of DST production: Multi-stage



# Event Combining

- Event combining is to combine data from other detectors using BCO(FULL)
  - The common method is applied for both (extended) trigger mode and streaming mode

- Method
  - Start with Reference BCOFULL (usually GL1)
  - Look for the hits having BCOFULL within “window” relative to Ref_BCOFULL




Link to macro

```
intt_sngl->SetNegativeBco(120-23);
intt_sngl->SetBcoRange(500);
```

# Convert

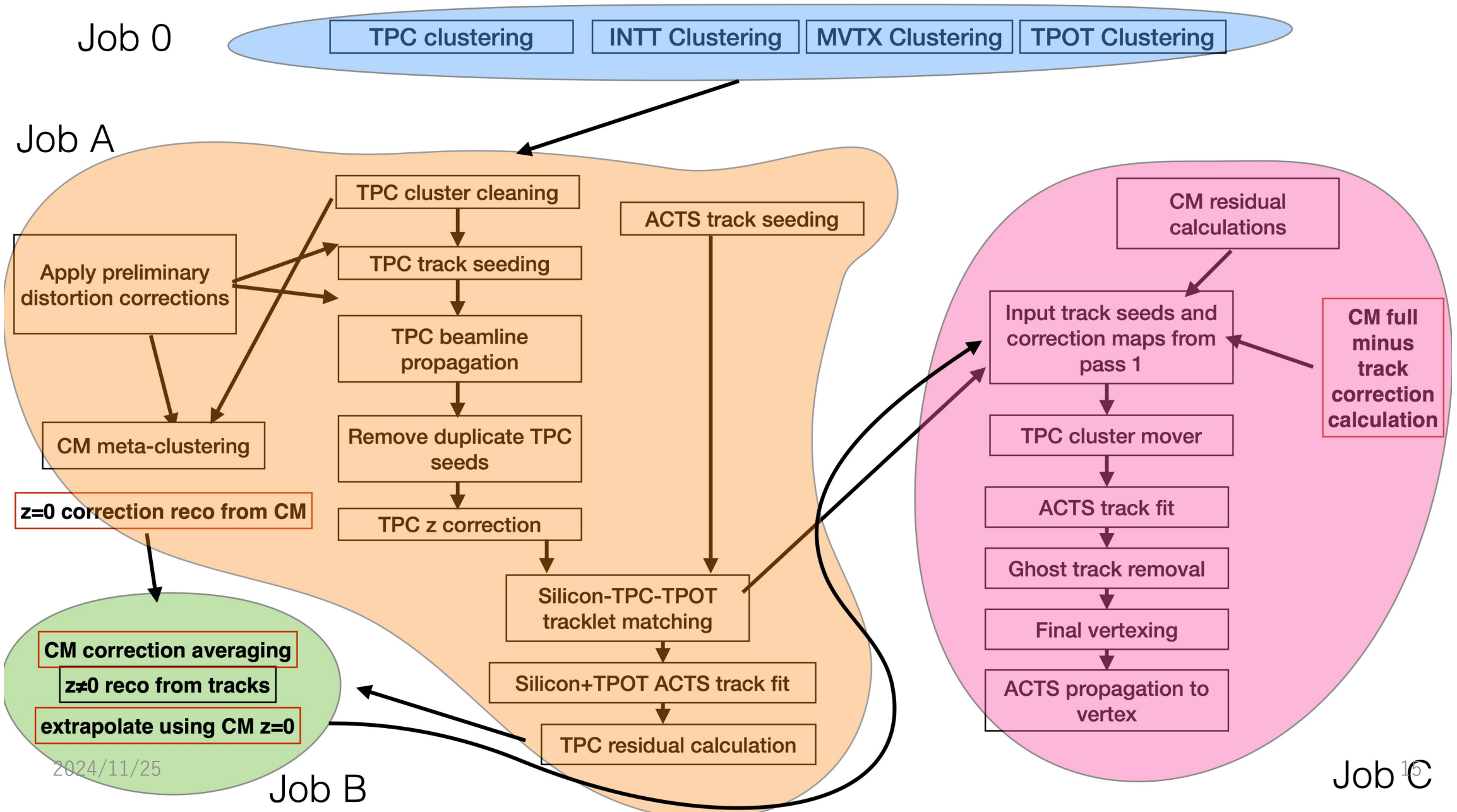
- Convert Ch in the detector(online) to Ch in sPHENIX software(offline)

- Convert BCO(7bits) to Time (Timebucket, 10bits)
  - Trigger: time = BCO – BCOFULL + offset
  - Stream: time = BCOFULL(stream) + BCO – GL1BCOFULL + offset

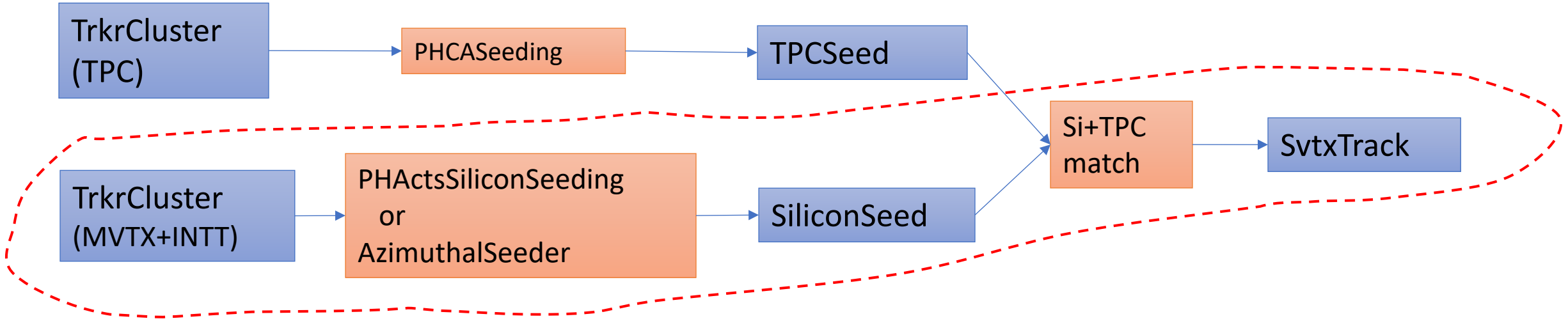
- Remove “BAD” channels using Hot/DeadMAP

- BCO Cut (optional)
  - BCO-BCOFULL cut was made for Run2023 analysis but probably no longer used for 2024-25 data since Fun4All prepared similar function (but not same)

# Detailed flow of track reconstruction

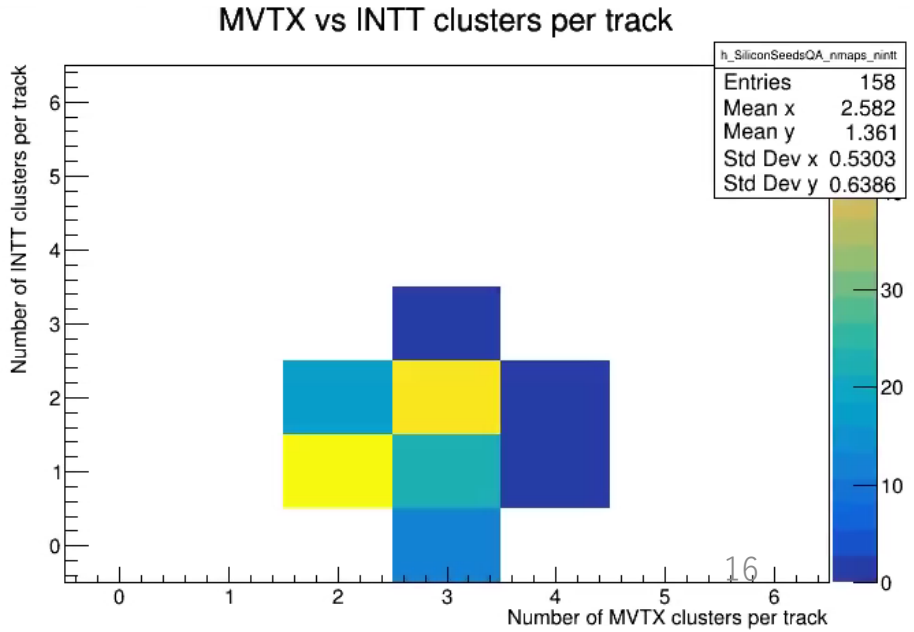


# Silicon Seed



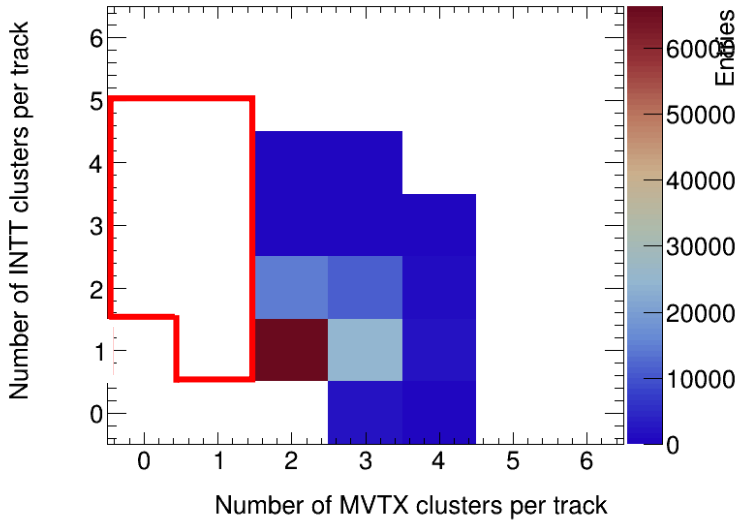

- https://github.com/sPHENIX-Collaboration/macros/blob/master/TrackingProduction/Fun4All_TrackAnalysis.C

- /gpfs/mnt/gpfs02/sphenix/user/hachiya/INTT/INTT/general_codes/hachiya/SiSeedAna/macro/Fun4All_TrackAnalysis.C



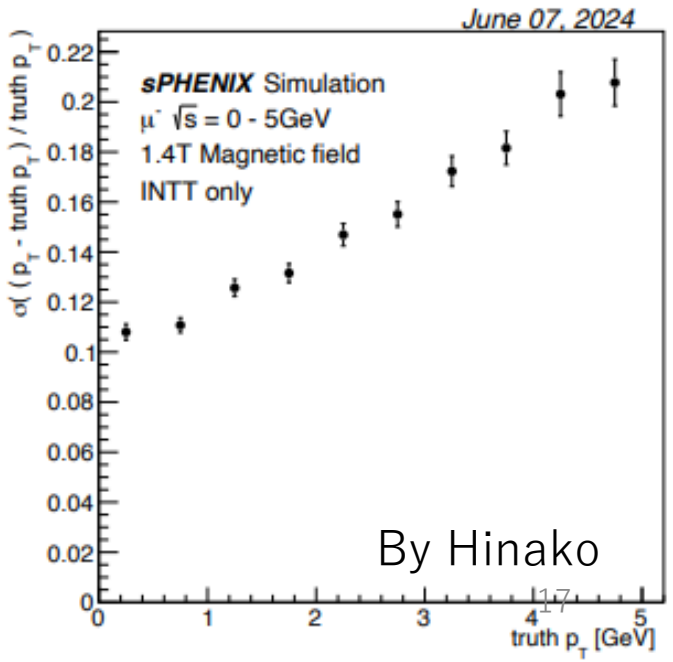
# Silicon seeding in tracking algorithm

- Tracking(Silicon Seeding) requires at least 2 MVTX + 1 INTT clusters
  - For low multiplicity such as p+p, this can be extended and relaxed.
  - S/N cannot be very good. Confirmation by outer detector (EMCAL) is needed.




- Furthermore, the momentum resolution can be improved by using EMCAL cluster
  - Help measuring Quarkonia by electron pairs

- Based on MC study by Hinako (NWU), INTT tracklet shows 10% pT resolution at pT=1GeV/c.



# Seed variable checking w/ offline QA code

- https://github.com/sPHENIX-Collaboration/coresoftware/blob/master/offline/QA/Tracking/SiliconSeedsQA.cc

- **SiliconSeedsQA.cc**
  - auto trackmap = findNode::getClass<SvtxTrackMap>(topNode, m_trackMapName);
  - SvtxTrack contents
    - virtual float get_p() const { return NAN; }
    - virtual float get_pt() const { return NAN; }
    - virtual float get_eta() const { return NAN; }
    - virtual float get_phi() const { return NAN; }

    - https://github.com/sPHENIX-Collaboration/coresoftware/blob/fb24f0a8d57e65c68ba0b442ac3a913b894e5207/offline/packages/trackbase_historic/SvtxTrack.h

# Results

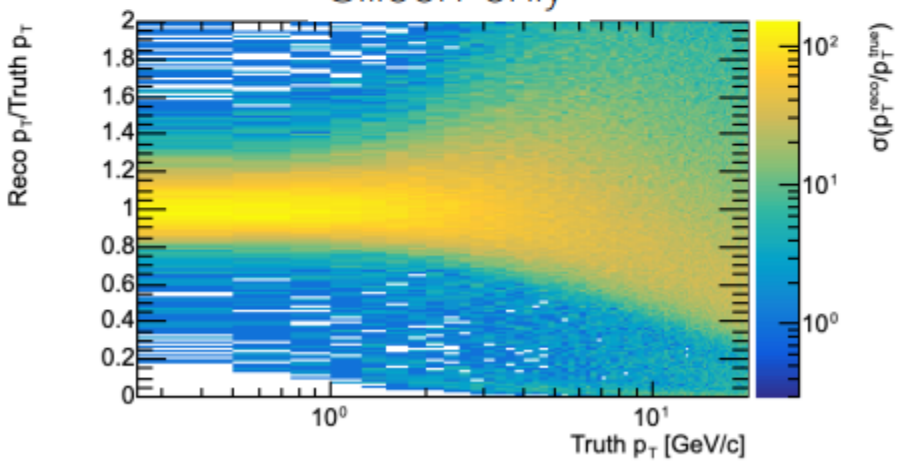
**“Reference” is QA file from Jenkins (full tracking with TPC, TPOT and silicon)**
**“New” is single pion tracking with silicon-only**

Thoughts

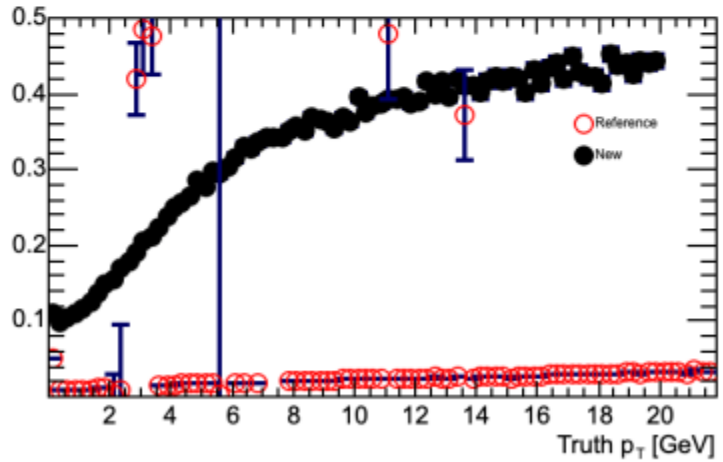
- Large divergence in reco to truth ratio (left) seems sensible at high pT where tracks are straighter so difficult to measure curvature
- Increase in momentum resolution (middle) without TPC seems sensible and gets to around 45%

Middle and right plots
Red circles – pion gun, full tracking (TPC+TPOT+Silicon)
Black circles – pion gun, silicon-only

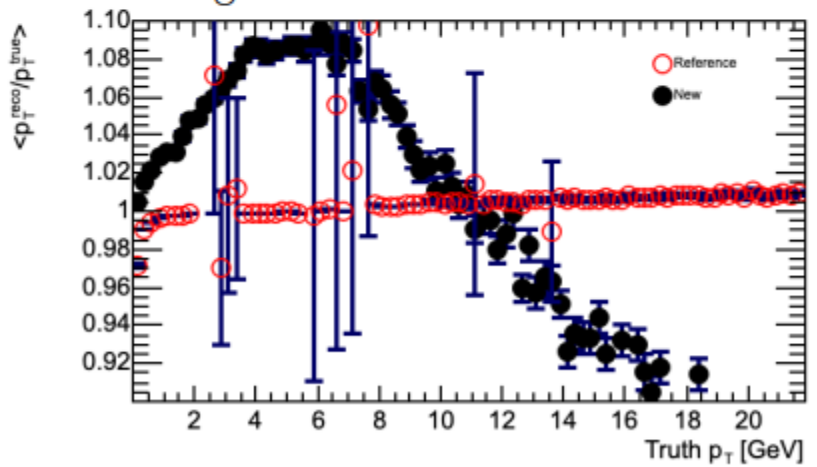
Silicon-only



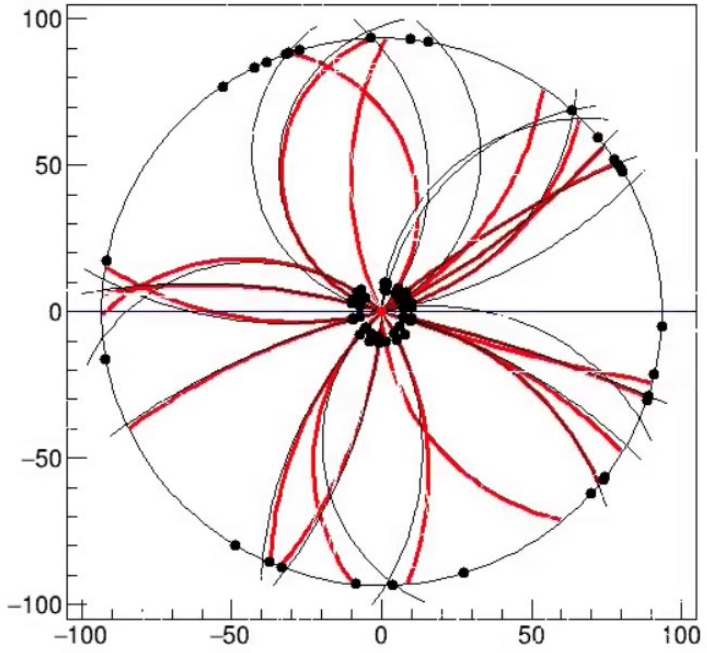

Momentum resolution

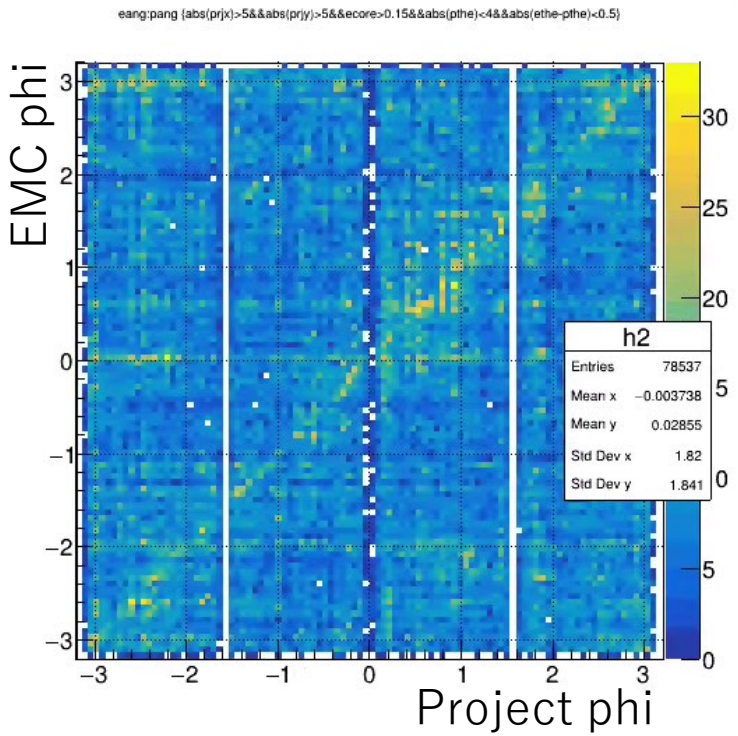


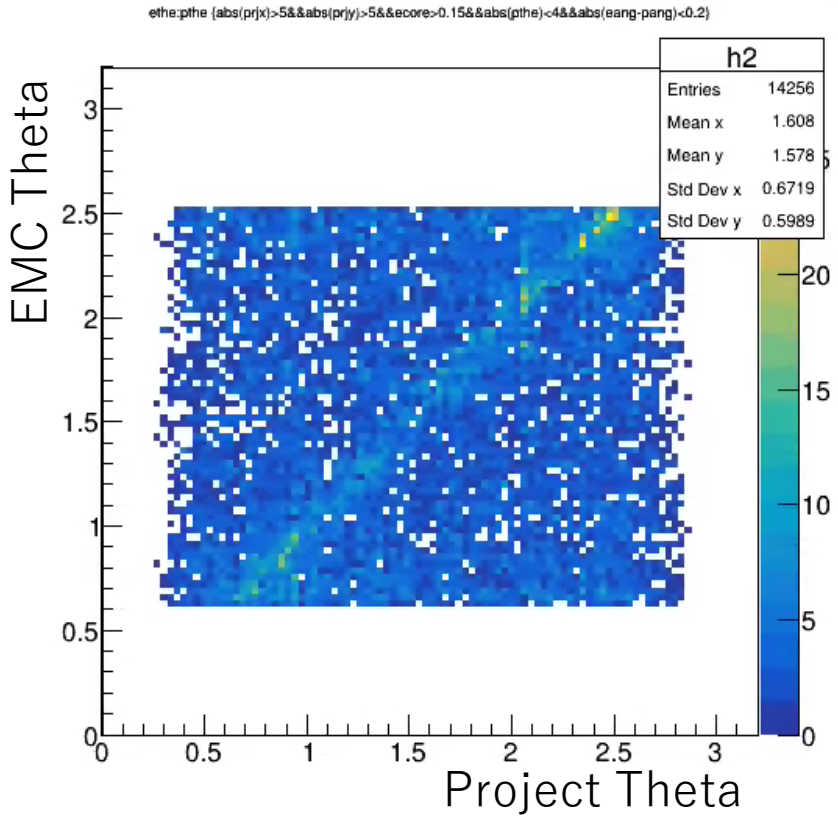

Average momentum reco/truth ratio



# Projection of INTT tracklet to EMCAL



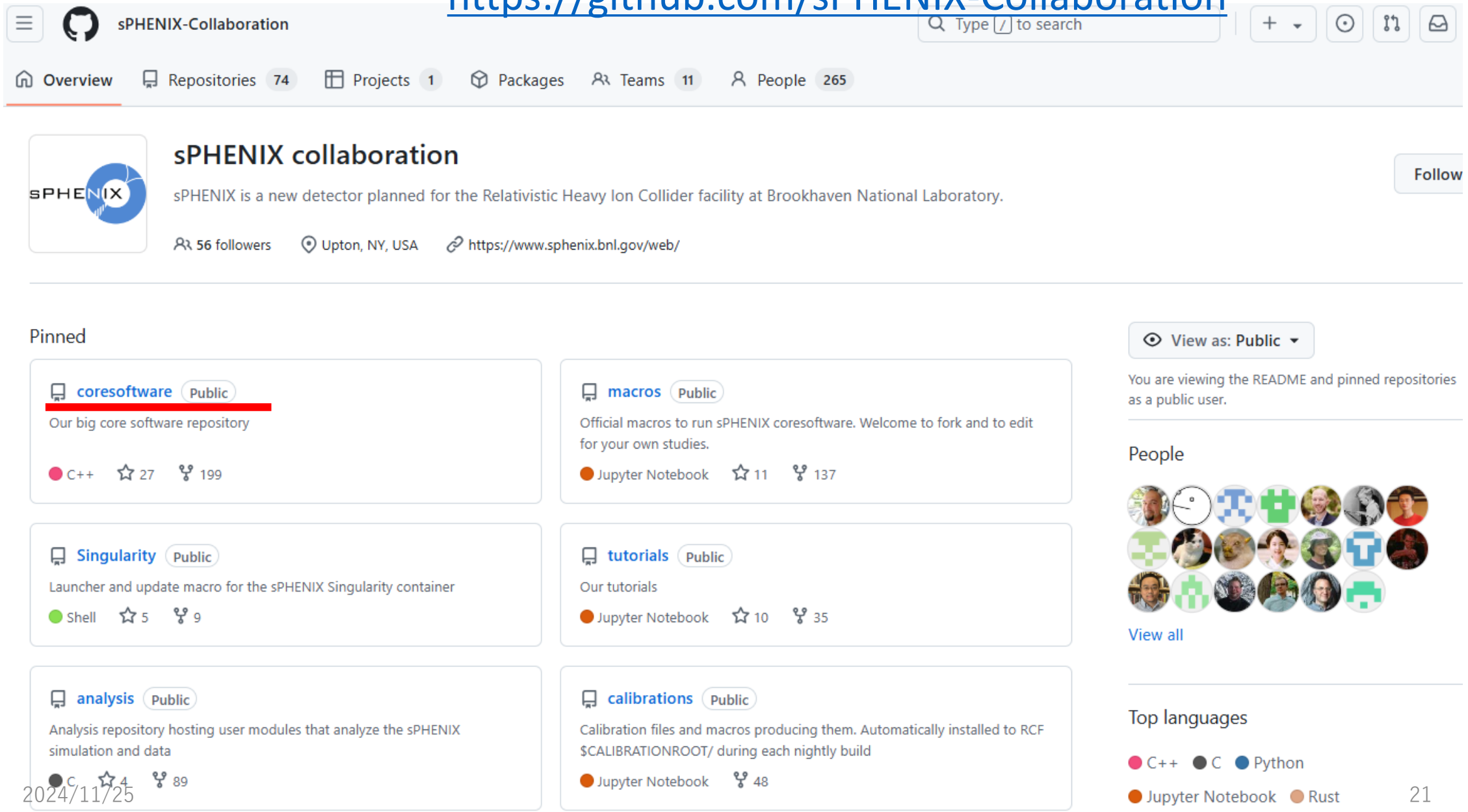



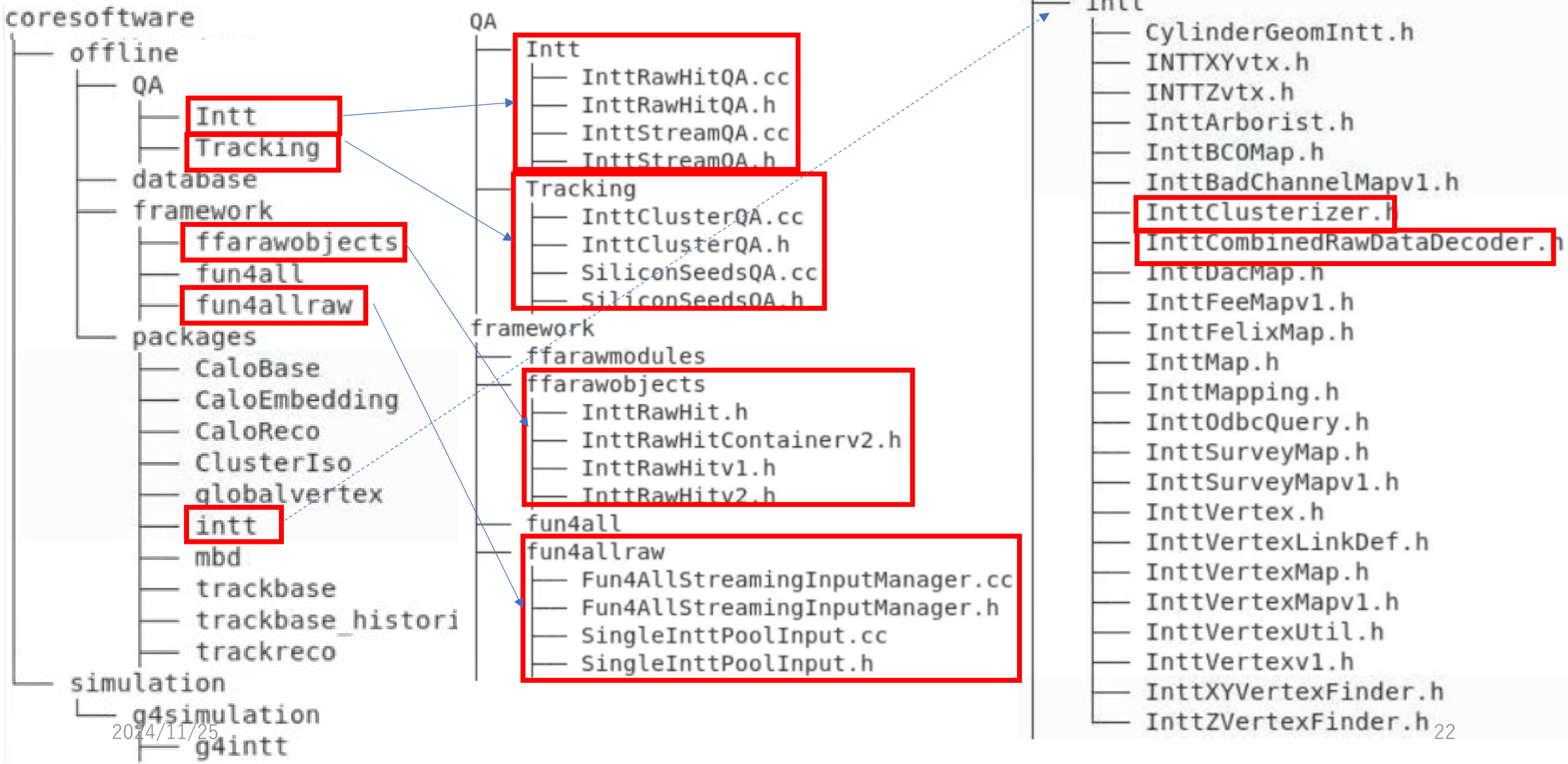

- MC(PYTHIA p+p MB)
  - Black line: INTT tracklet, Red: MC truth track
- INTT Tracklet is projected to EMCAL surface and search for the closest EMC cluster
  - INTT tracklet : 2 INTT clusters + XY vertex
  - Simple circle projection
- INTT + EMC matching works in p+p. Can be improved with MVTX

# sPHENIX software on GitHub

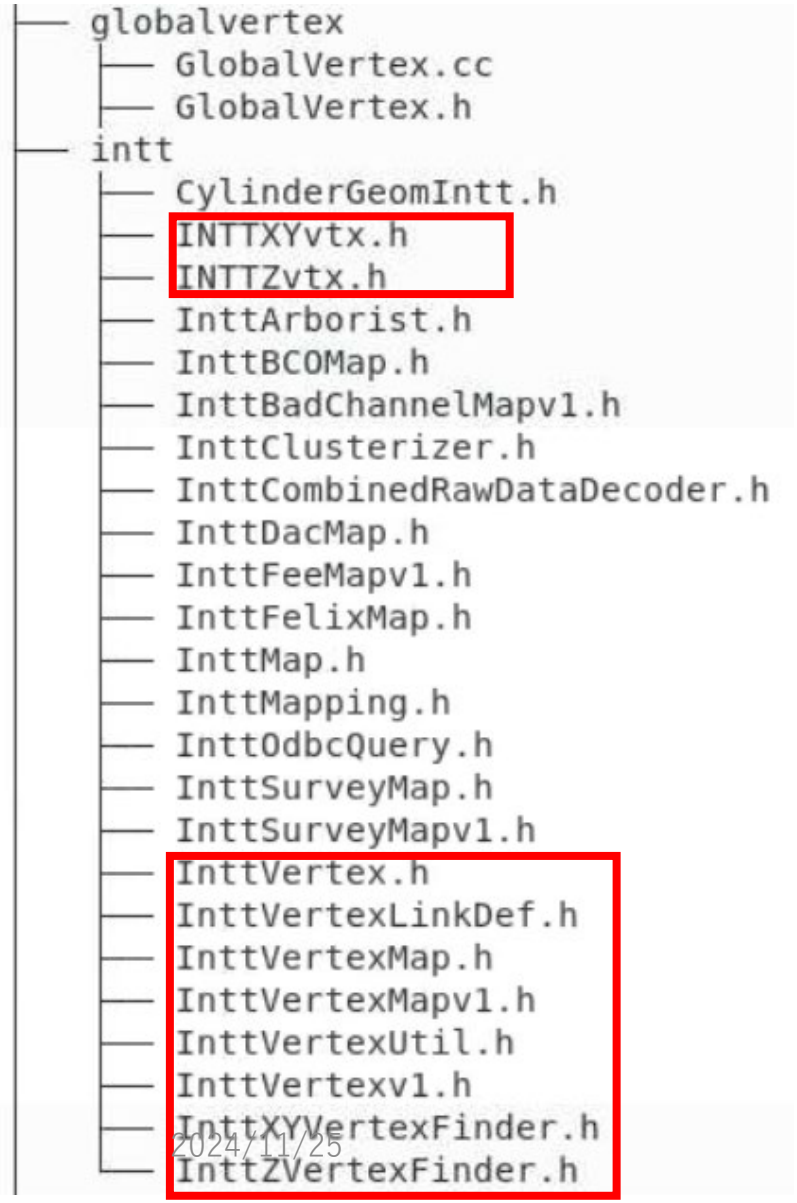
https://github.com/sPHENIX-Collaboration



# Coresoftware in github



# INTT vertex

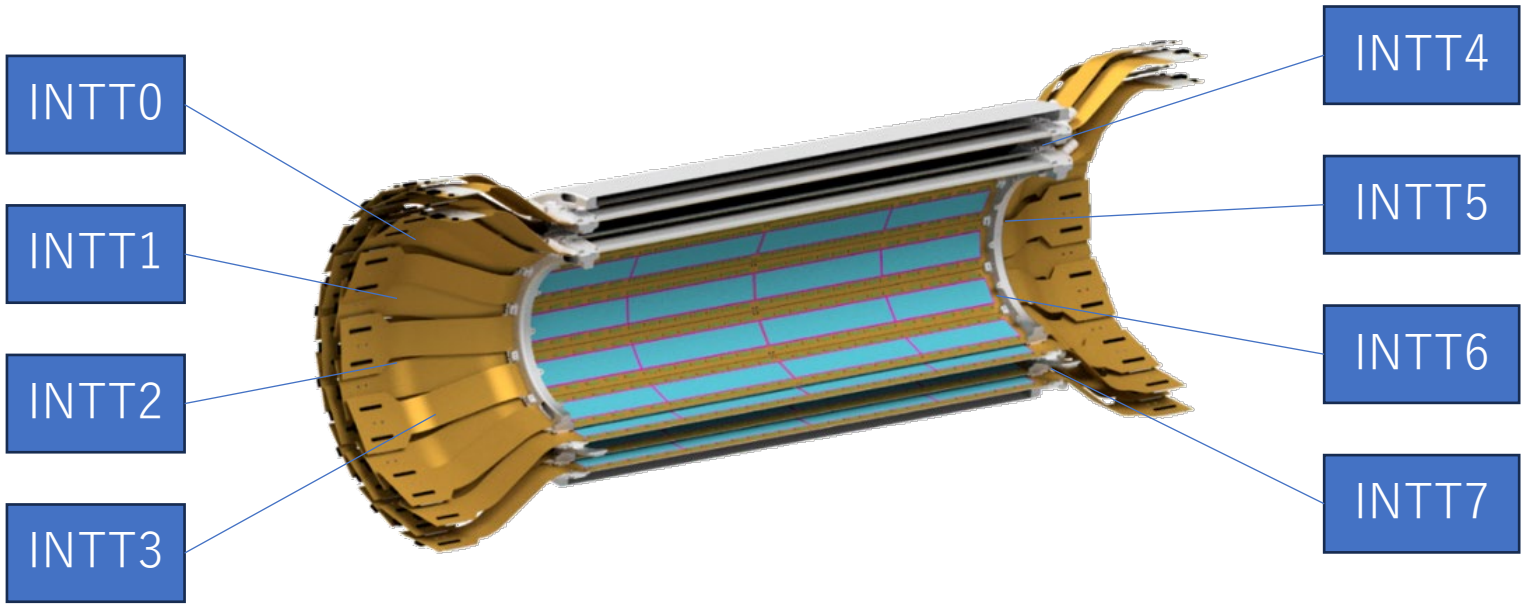


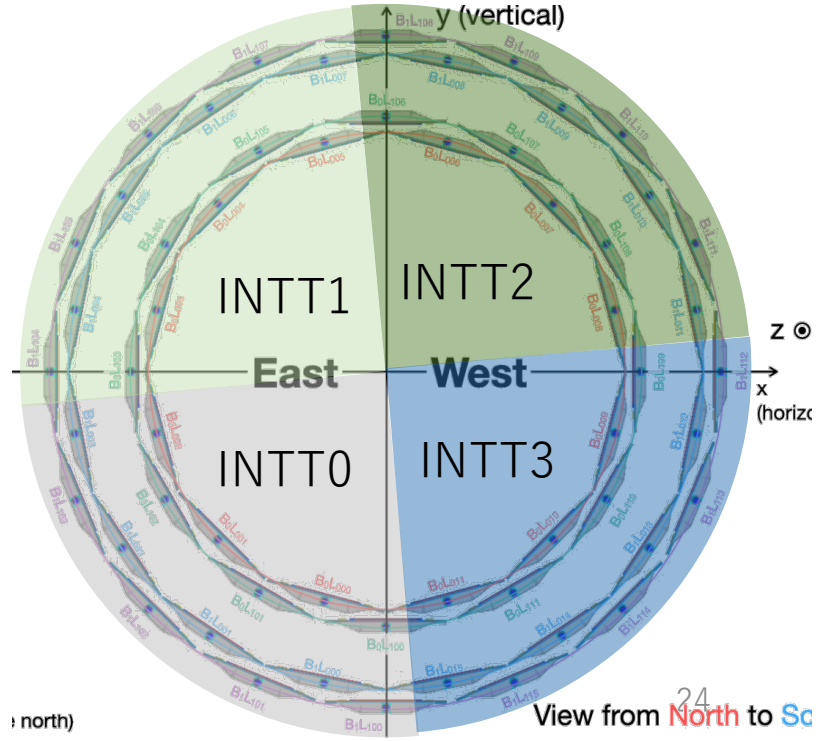

- F4A module to calculate INTT XY and Z vertex available in sPHENIX-GITHUB
- These modules developed based on ChengWei's vertexing code
- It works but need to separate clusters by time (BCO)

# Definition of numbering

- Two definitions of chip/channel in Detector(ONLINE) / OFFLINE
- Three different definitions of ladder ID
  - Detector(FELIX/ONLINE)/OFFLINE

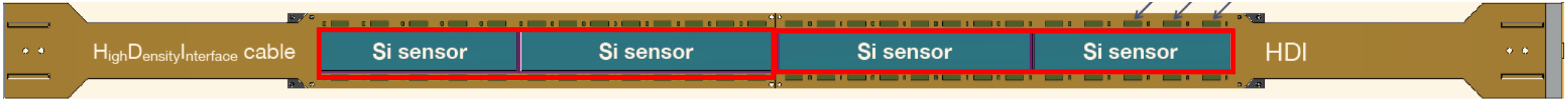
- INTT uses 8 FELIXs. One FELIX covers a quadrant with 14 ladders connected



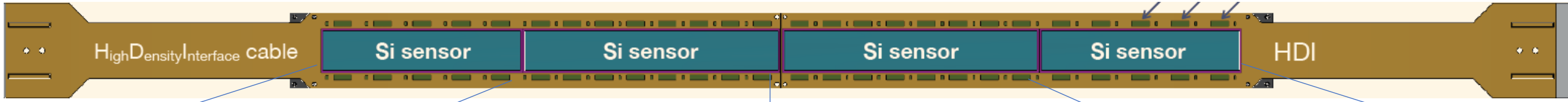



# Detector channel vs Offline Channel

| | Detector Channel | Offline Channel |
|---|---|---|
| **Ladder Position** | • FELIX_ID<br>• Ladder | • Layer<br>• Ladder phi |
| **Channel Position in ladder** | • Chip<br>• Channel | • Ladder Z<br>• Row<br>• Col |
| | Half Ladder | Full Ladder |



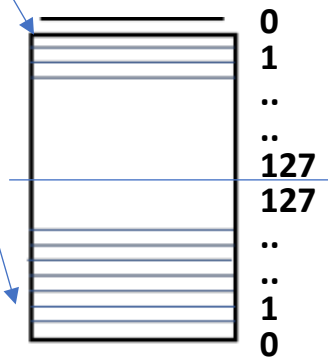
# Channel ID  detector vs offline




detector

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 |

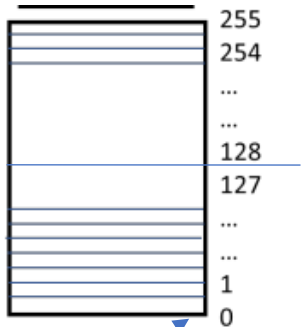
| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |




Detector channel

Chip (26 chips)

Channel (0-127)




Offline Channel

Ladder Z

Col (chip in sensor)

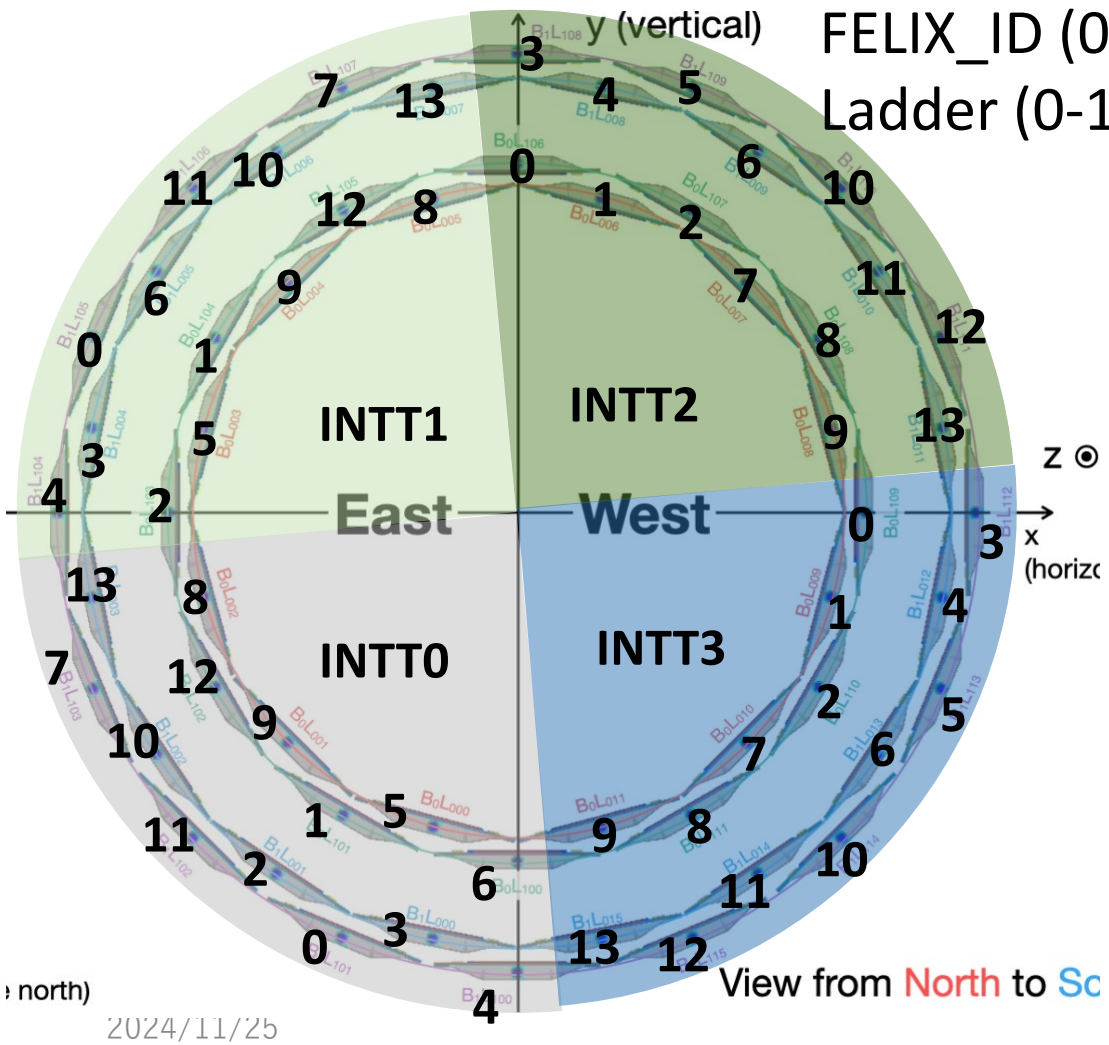
Row (0-255)

Offline

| Ladder-Z = 1 | | | | | Ladder-Z = 0 | | | | | | | | Ladder-Z = 2 | | | | | | | | Ladder-Z = 3 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 |

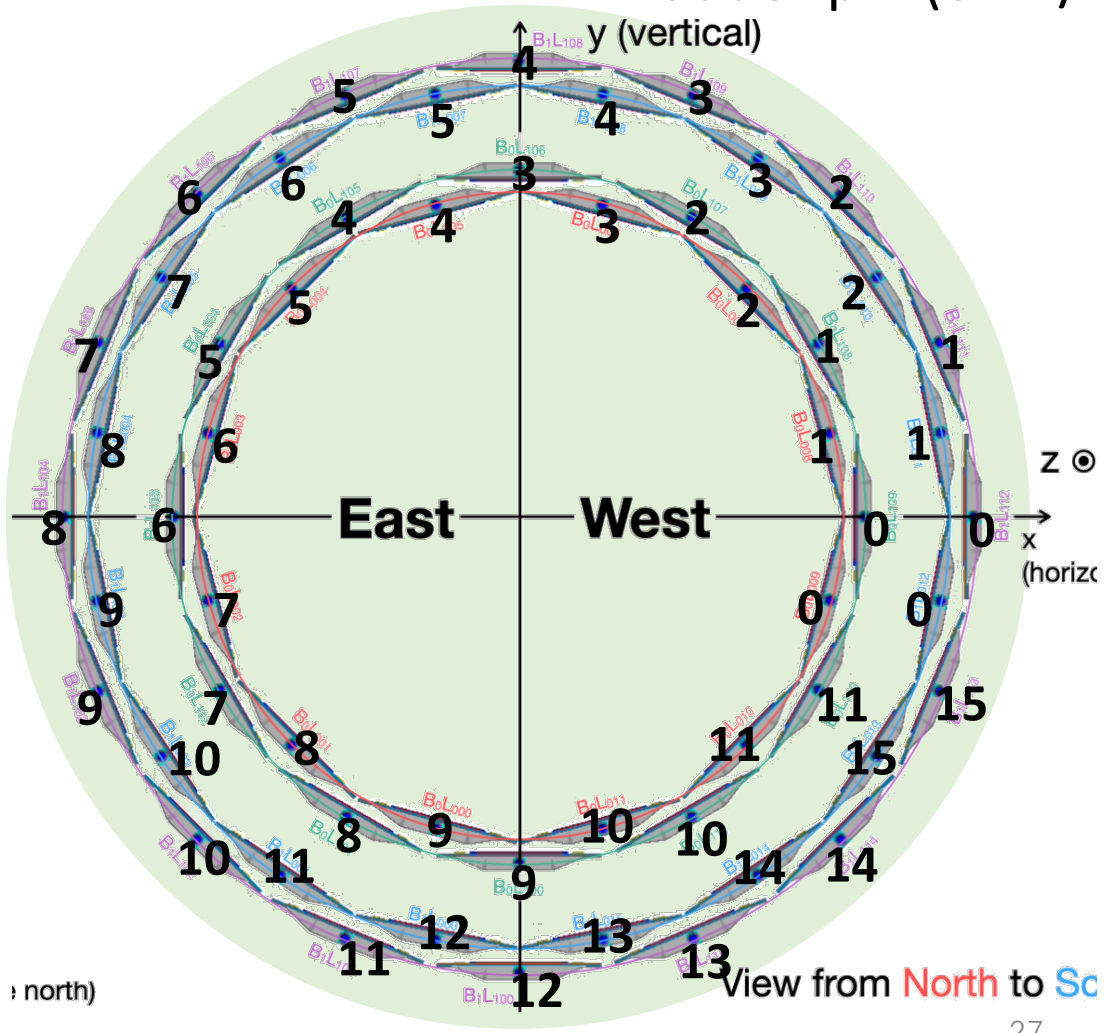
# Ladder ID

Detector
(depends on how the ladder connected)



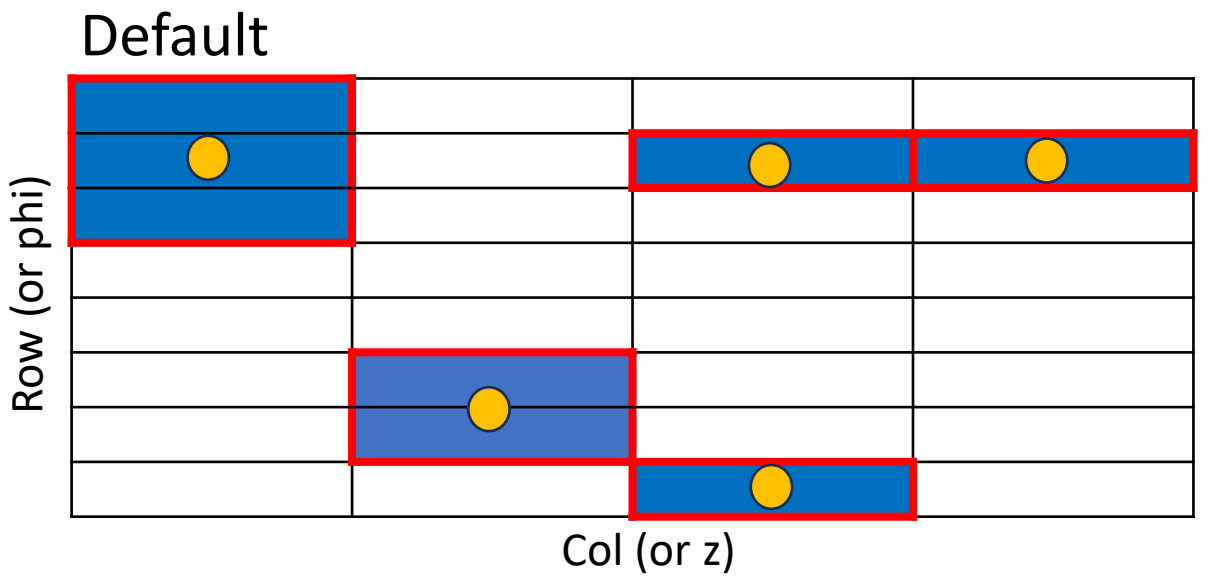

OFFLINE level

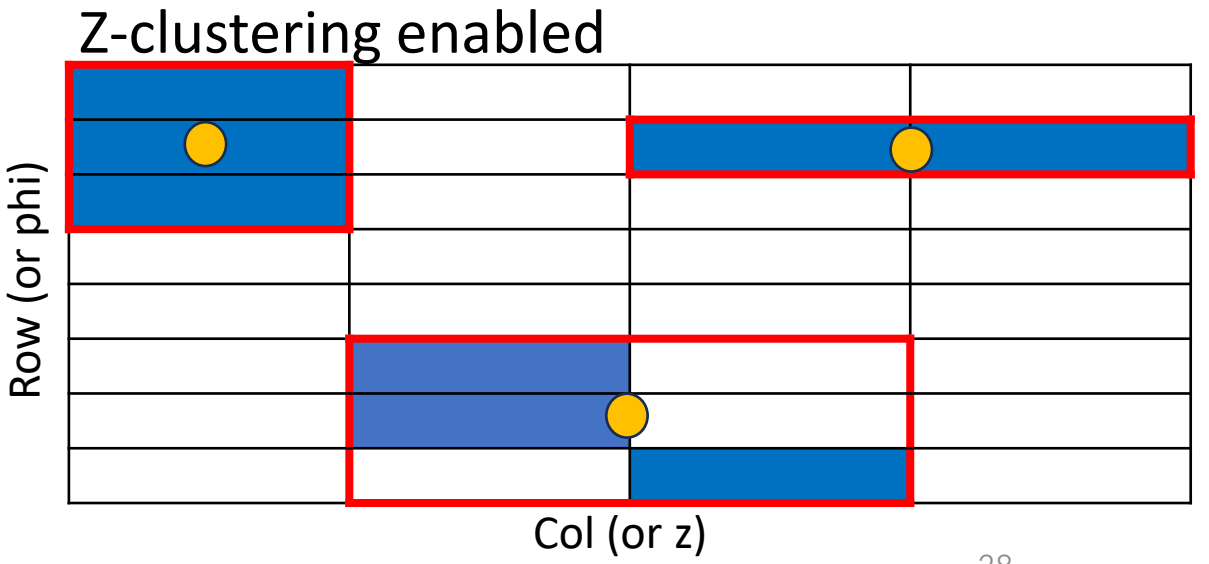
Layer (3,4,5,6)
Ladder phi (0-11/15)



# Clustering

- Adjacent hits is grouped to a “cluster”
- Definition of “adjacent” is
  - Row(phi) diff <=1 & Col(z) diff<=1
- Clustering options
  - Z-clustering (=No Z clustering as default)
  - ADC-weight (=No ADC weight as default)



# InttClusterizer Code

## Adjacent hits in InttClustering.C

```
bool InttClusterizer::ladder_are_adjacent(const std::pair<TrkrDefs::hitkey, TrkrHit
{
  if (get_z_clustering(layer))
  {
    if (fabs(InttDefs::getCol(lhs.first) - InttDefs::getCol(rhs.first)) <= 1)
    {
      if (fabs(InttDefs::getRow(lhs.first) - InttDefs::getRow(rhs.first)) <= 1)
      {
        return true;
      }
    }
  }
  else if (fabs(InttDefs::getCol(lhs.first) - InttDefs::getCol(rhs.first)) == 0)
  {
    if (fabs(InttDefs::getRow(lhs.first) - InttDefs::getRow(rhs.first)) <= 1)
    {
      return true;
    }
  }

  return false;
}
```

## ADC weight in InttClustering.C

```
if (_make_e_weights[layer])
{
  cluslocaly = ylocalsum / (double) clus_adc;
  cluslocalz = zlocalsum / (double) clus_adc;
}
else
{
  cluslocaly = ylocalsum / nhits;
  cluslocalz = zlocalsum / nhits;
}
```

## Trkr_Clustering.C (production macro)

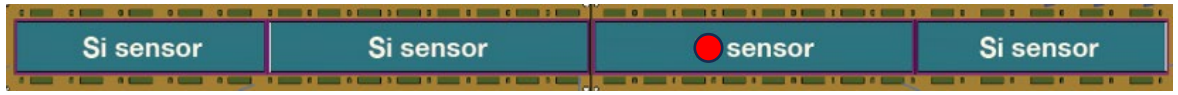
```
void Intt_Clustering()
{
  int verbosity = std::max(Enable::VERBOSITY, Enable::INTT_VERBOSITY);
  Fun4AllServer* se = Fun4AllServer::instance();

  InttClusterizer* inttclusterizer = new InttClusterizer("InttClusterizer", G4MVTX::n_maps_l
  inttclusterizer->Verbosity(verbosity);
  // no Z clustering for Intt type 1 layers (we DO want Z clustering for type 0 layers)
  // turning off phi clustering for type 0 layers is not necessary, there is only one strip
  // per sensor in phi
  for (int i = G4MVTX::n_maps_layer; i < G4MVTX::n_maps_layer + G4INTT::n_intt_layer; i++)
  {
    if (G4INTT::laddertype[i - G4MVTX::n_maps_layer] == PHG4InttDefs::SEGMENTATION_PHI)
    {
      inttclusterizer->set_z_clustering(i, false);
    }
  }
  se->registerSubsystem(inttclusterizer);
}
```

# Geometry conversion method

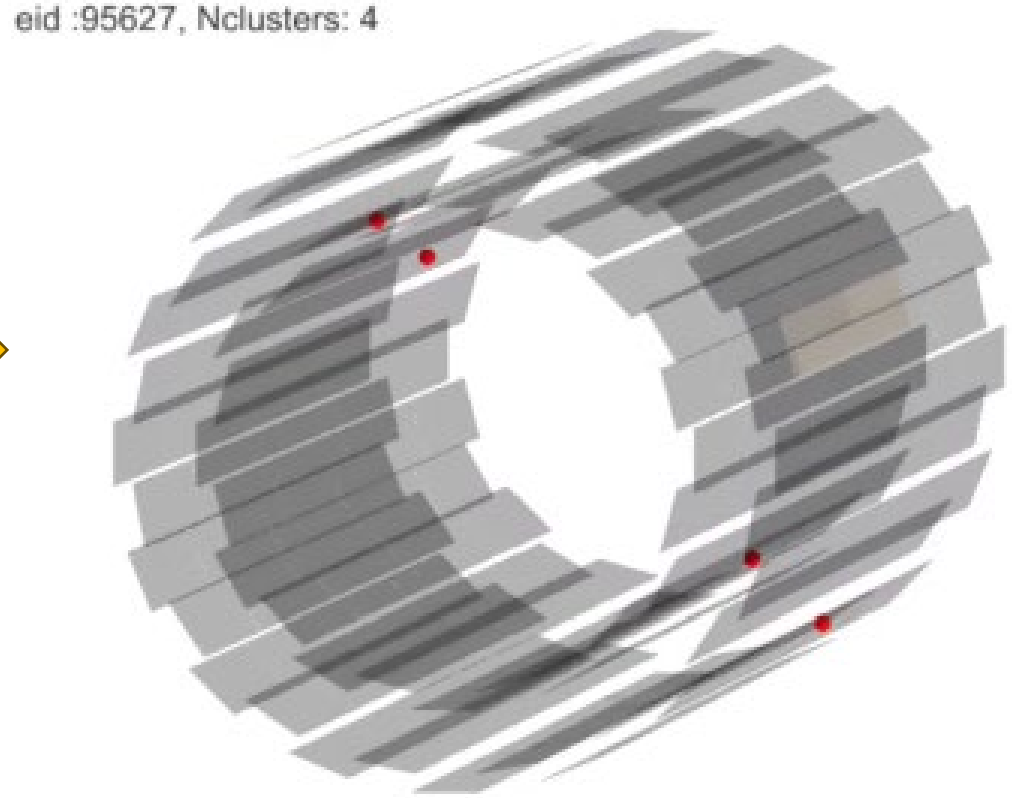
- ACTSGeomtry is sPHENIX geometry model
  - Geometry model has the detector location and provides the conversion rule from the local frame to the global 3D position in sPHENIX frame

Local frame (ladder frame)




Convert

global frame with 3D



# Geometry conversion method

- ACTSGeomtry is sPHENIX geometry model
  - Detector geometry (ladder/channel positions) is based on GEANT and is built on-the-fly when running DST reconstruction
    - Therefore, it takes longer time to construct the full sPHENIX detector in memory.
    - To avoid this longer processing time, sPHENIX software uses “pre-built” detector model loaded from the file

Traditional way to build the detector model in memory

```
// Initialize the selected subsystems
G4Init();
G4Setup();
ACTSGEOM::ActsGeomInit();
```

Takes longer time

More quick way by loading the detector model from the file

```
Enable::CDB = true;
rc->set_StringFlag("CDB_GLOBALTAG", "ProdA_2024");
rc->set_uint64Flag("TIMESTAMP", runnumber);
std::string geofile = CDBInterface::instance()->getUrl("Tracking_Geometry");

Fun4AllRunNodeInputManager *ingeo = new Fun4AllRunNodeInputManager("GeoIn");
ingeo->AddFile(geofile);
se->registerInputManager(ingeo);
G4MAGNET::magfield_rescale = 1;
ACTSGEOM::ActsGeomInit();
```

# Which data is good for your analysis

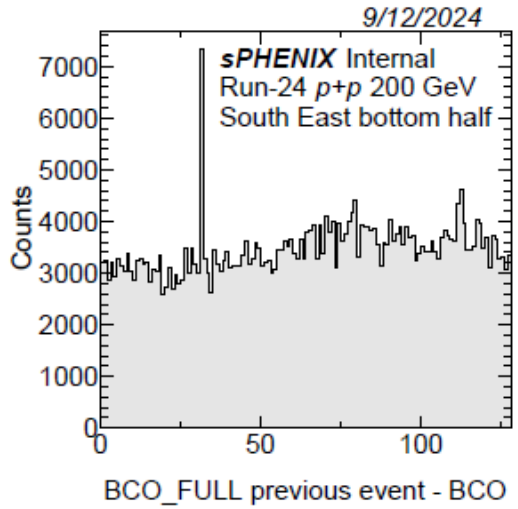
Raw data → InttRawhit → TrkHitSet → Cluster

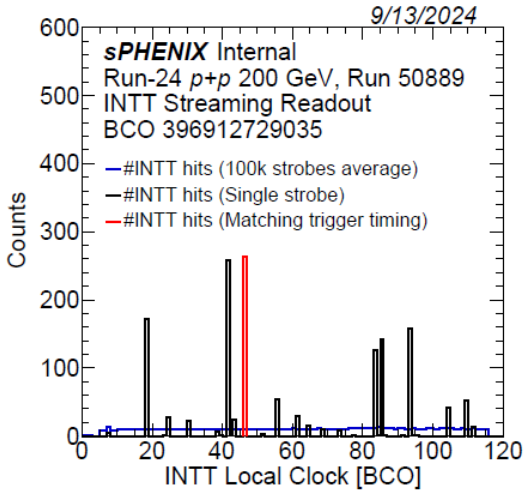
EVT file | DST file | DST file | DST file

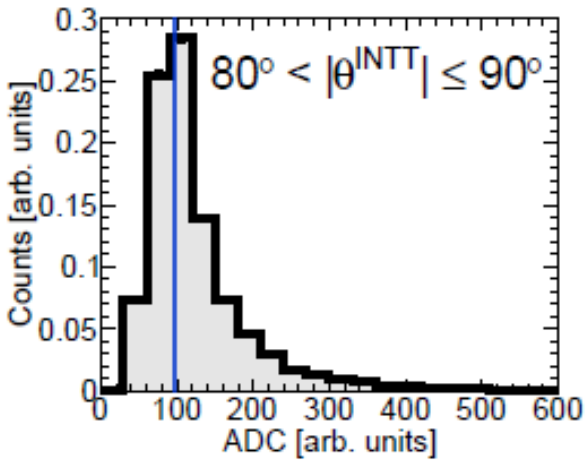
- Raw data: develop/debug the event combine
  (not good for everyone)

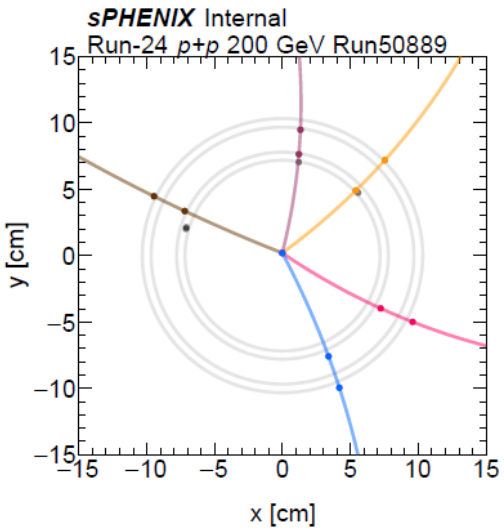
- InttRawhit: Quantities related to Hardware,
  - BCO, Mixup, Hot/dead, streaming QA

- Cluster : Quantities related to PHYSICS,
  - ADC for MIP, 3D position for further analyses

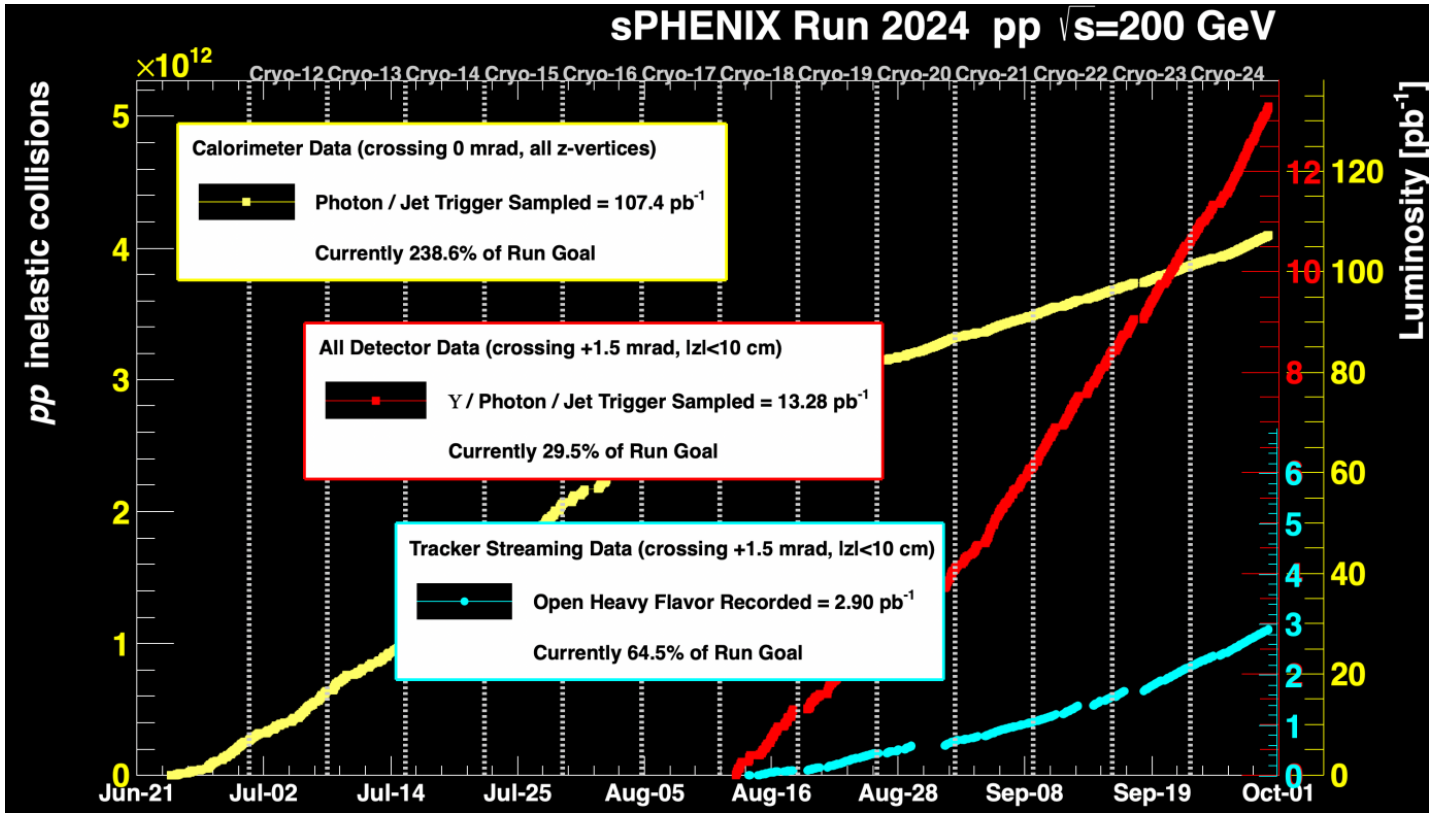


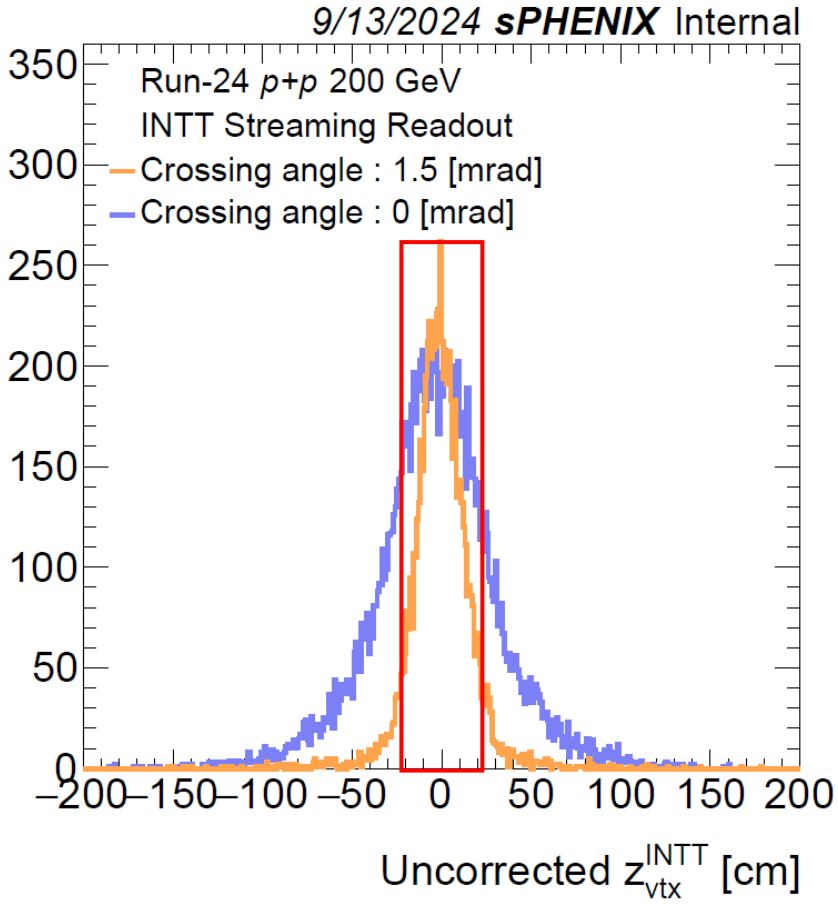



# Toward Quarkonia measurement in pp200GeV



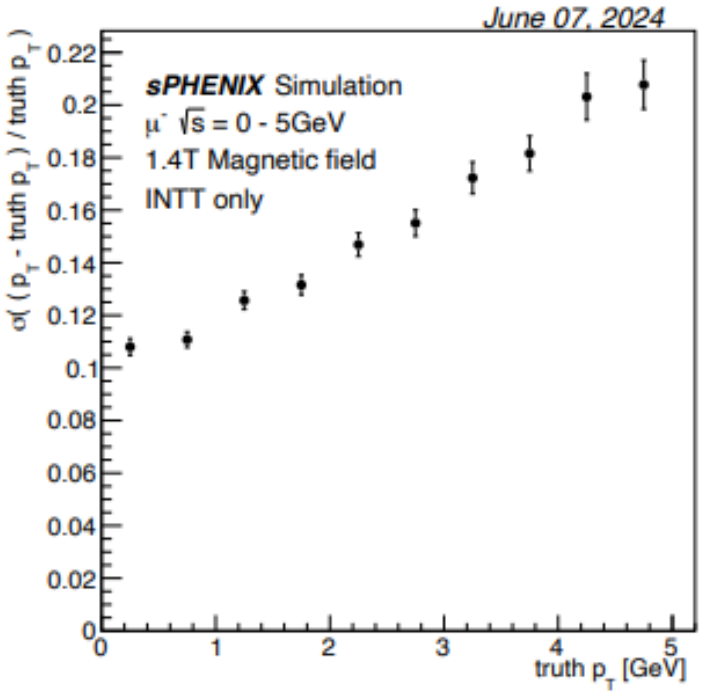




- PP-CALO data is 120pb-1
- INTT took data with CALO system for entire PP data
  - can be good for ~50% of PP-CALO data (consistent w/ BUP luminosity)

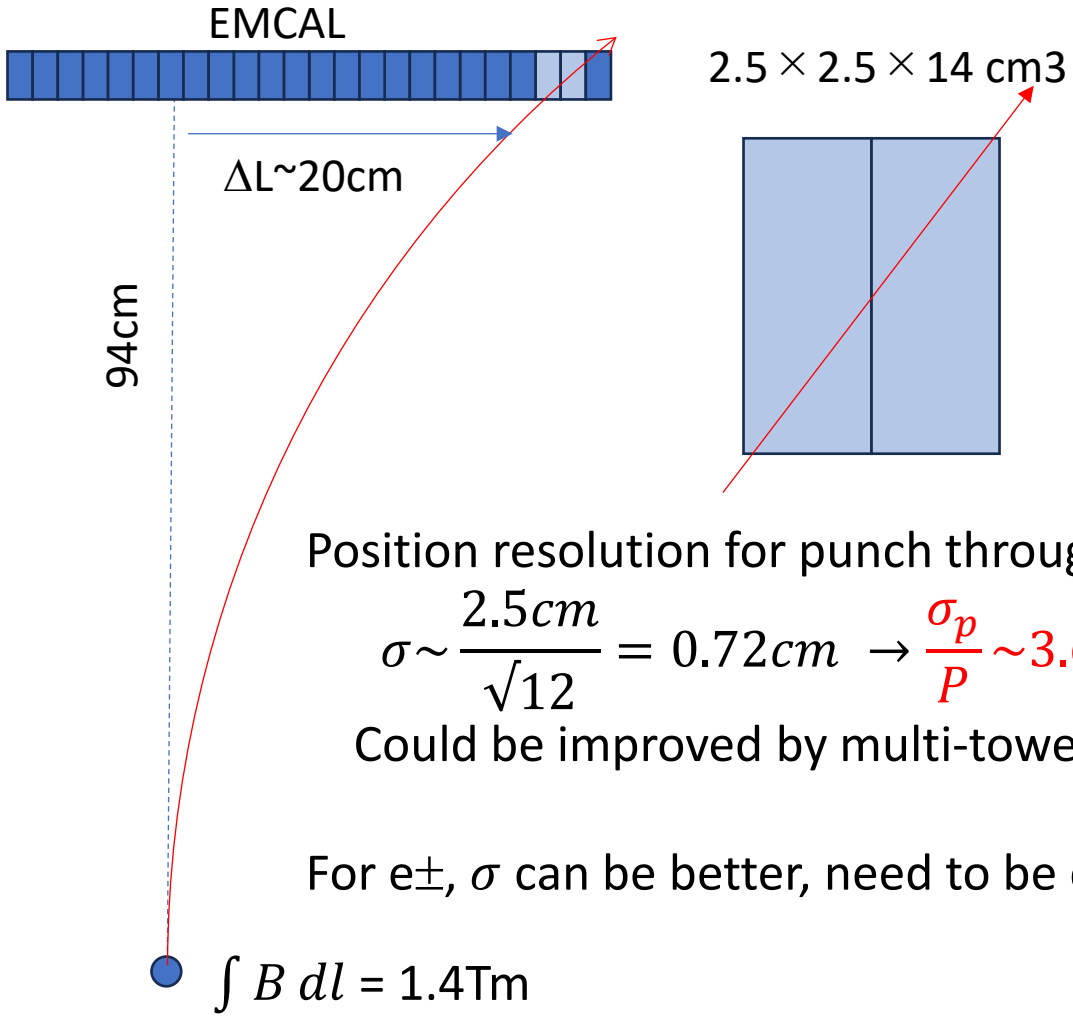
Chance for quarkonia measurements with Si + EMC tracking

# Si + EMC tracking

INTT pT resolution ~ 10% @ 1GeV/c




P resolution get better with EMC




Position resolution for punch through hadrons

$$\sigma \sim \frac{2.5cm}{\sqrt{12}} = 0.72cm \rightarrow \frac{\sigma_p}{P} \sim 3.6\%$$

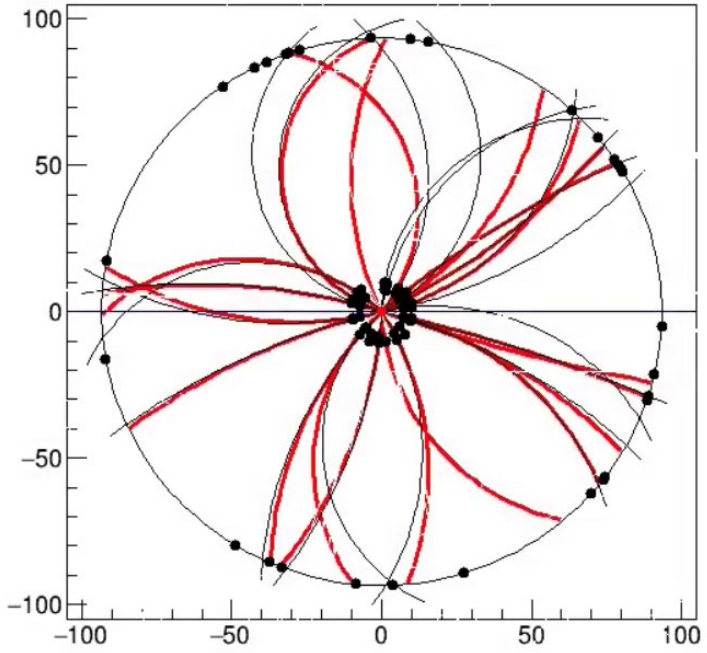
Could be improved by multi-towers w/ E weight

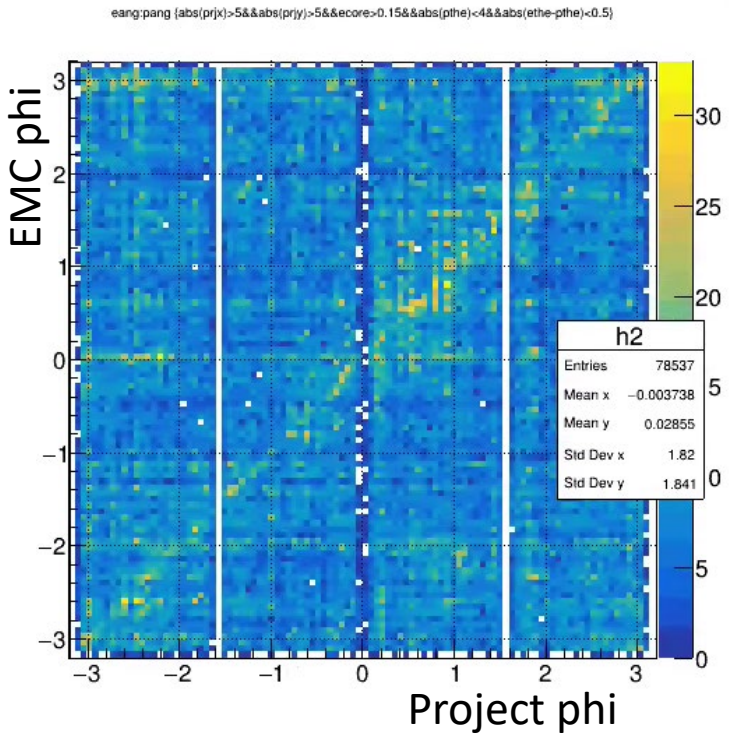
For e±, $\sigma$ can be better, need to be checked by MC

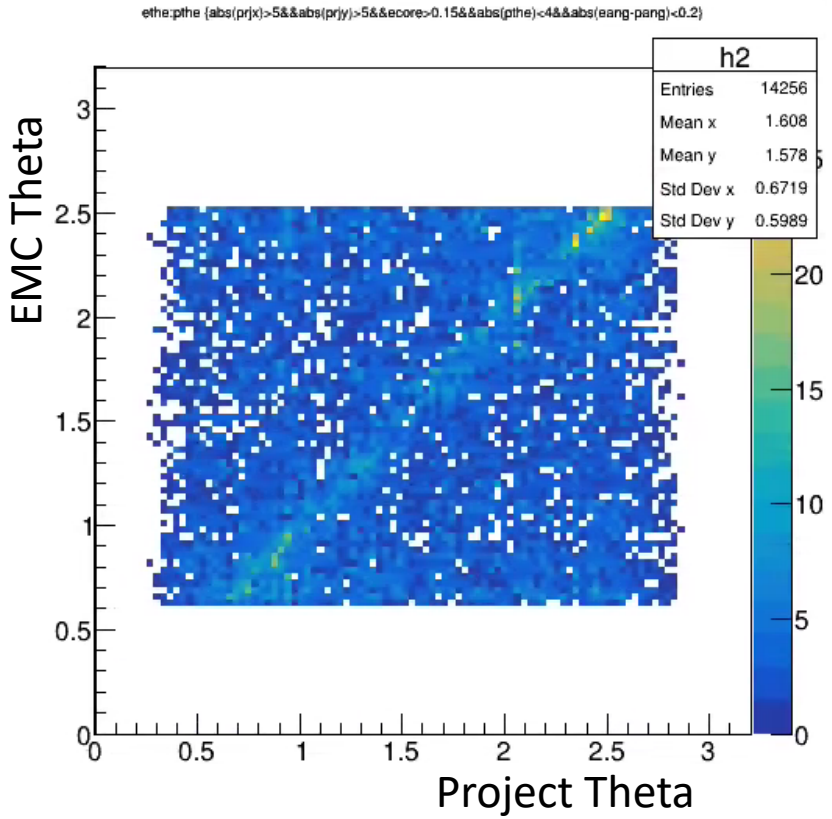
$\int B\, dl$ = 1.4Tm

$pT_{kick}$ = 0.4GeV/c → ΔL ~ 20cm for 1GeV/c particle

# Test with INTT + EMCAL tracking



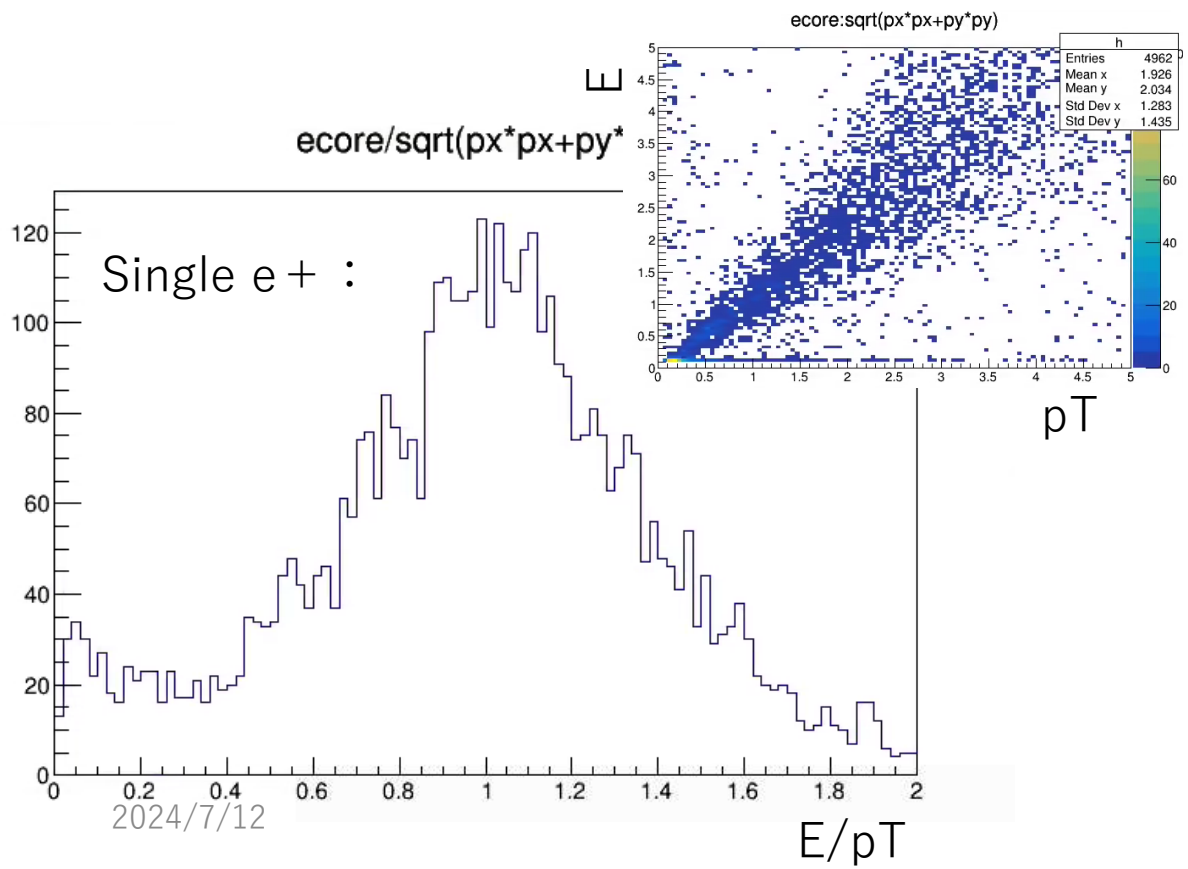


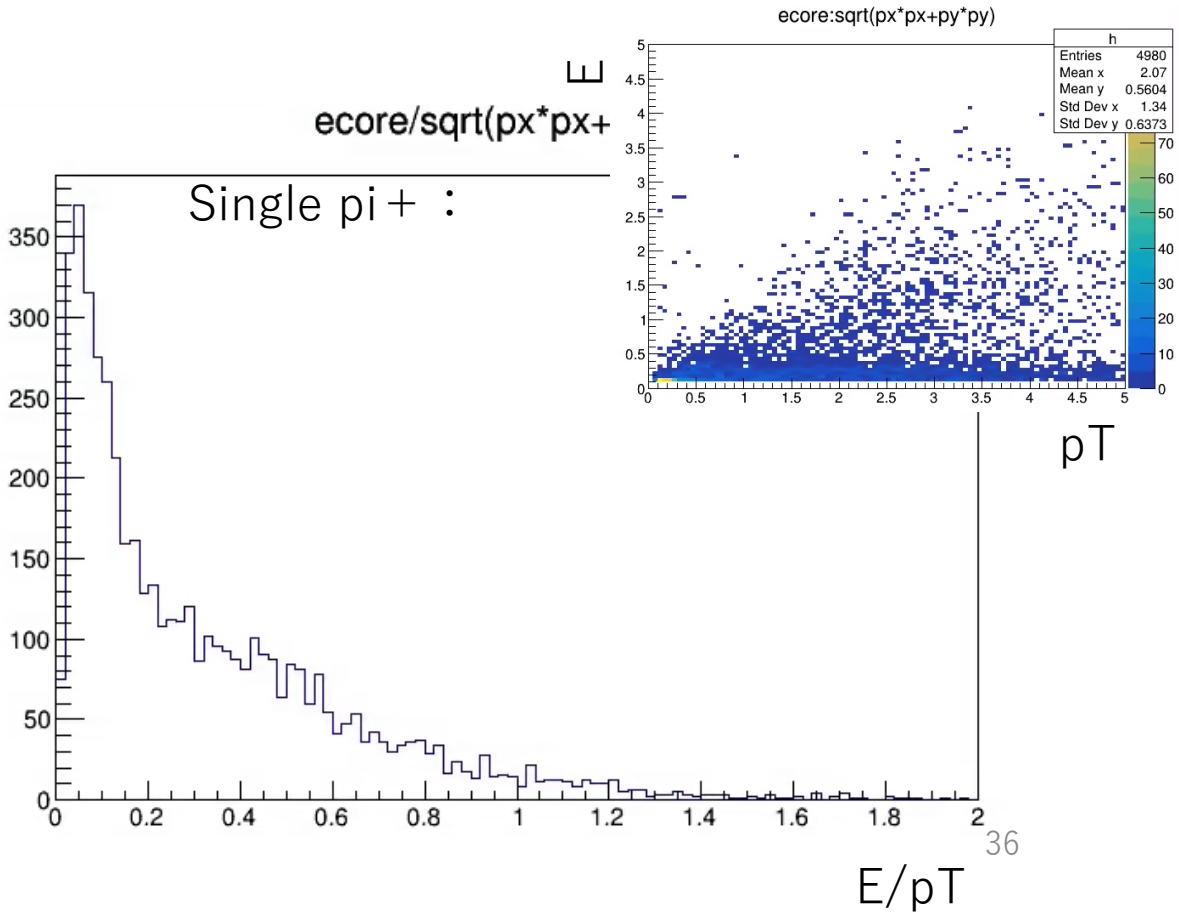




- MC(PYTHIA p+p MB)
  - Black line: INTT tracklet, Red: MC truth track
- INTT Tracklet is projected to EMCAL surface and search for the closest EMC cluster
  - INTT tracklet : 2 INTT clusters + XY vertex
  - Simple circle projection
- INTT + EMC matching works in p+p. Can be improved with MVTX

# Electron ID by E/p in single e+ and pi+ MC

- E/p is eID quantity (mom not improved by EMCAL)
  - E/p ~ 1 for e+ (E and p should be matched),
  - but <<1 for pi+ (E in EMCAL for pi+ is E(MIP~0.2GeV constant)
- If p reso get better,  E/p peak gets shaper for electrons



# INTT tracklet efficiency by Hinako

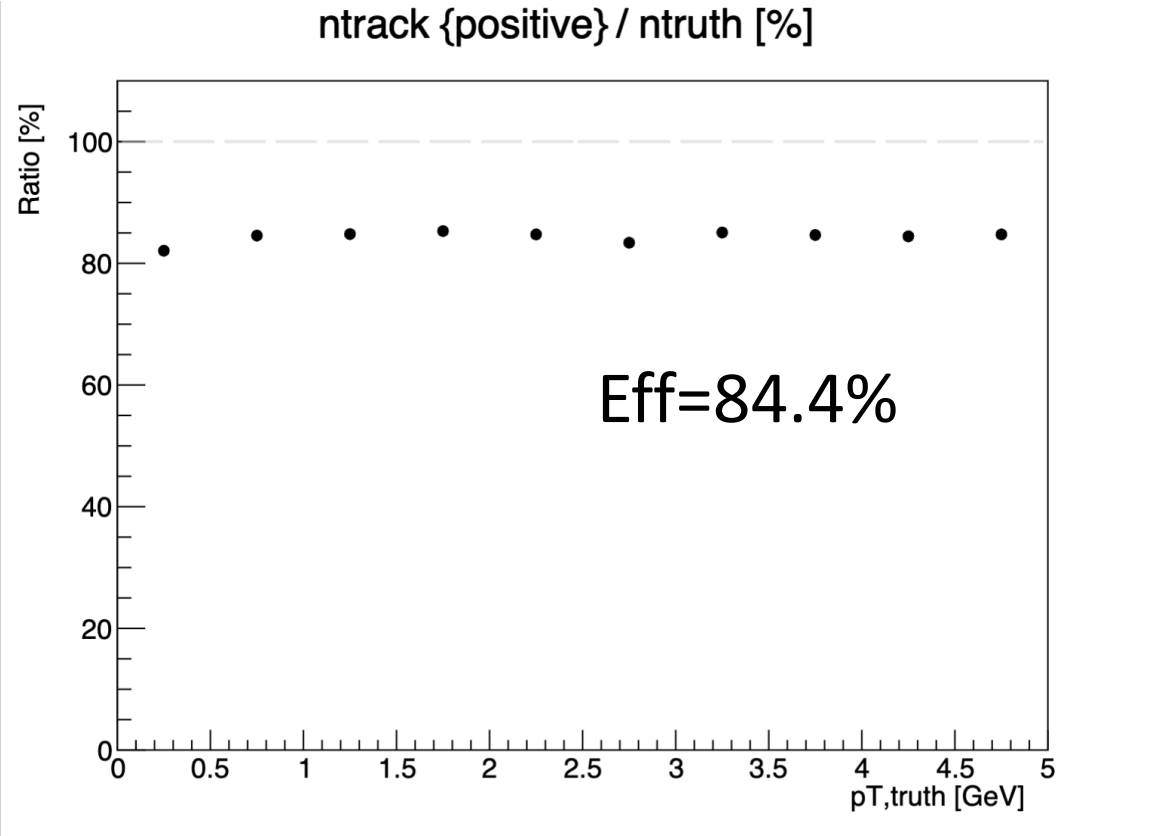
Hinako is studying the reconstruction efficiency of INTT tracklets

- $Eff = \frac{N_{reco}}{N_{truth}} = 84.4\ \%$

- Seems lower than expected value?

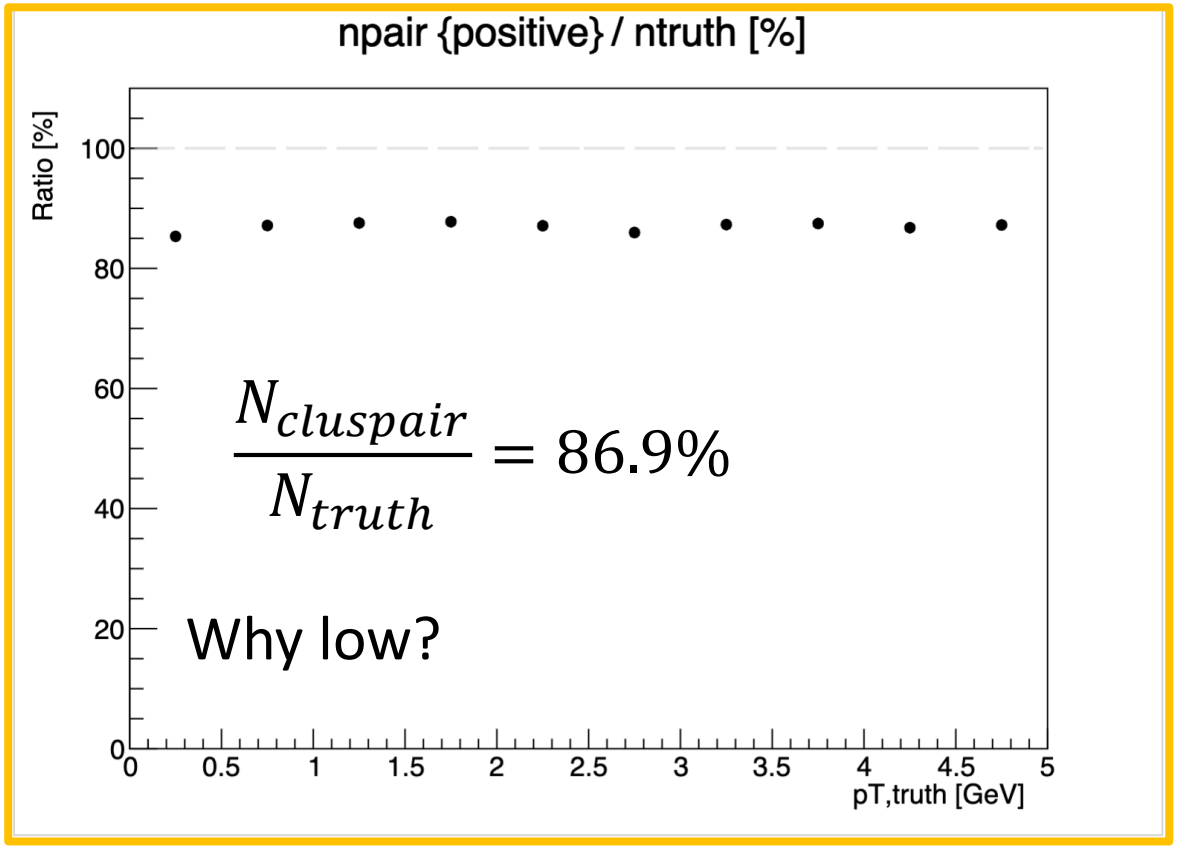
【MC】

- single muon with 0-5 flat pT
- The # of events : 100K events
- Magnetic field : B-off
- vertex : Gaussian with +- 1cm sigma



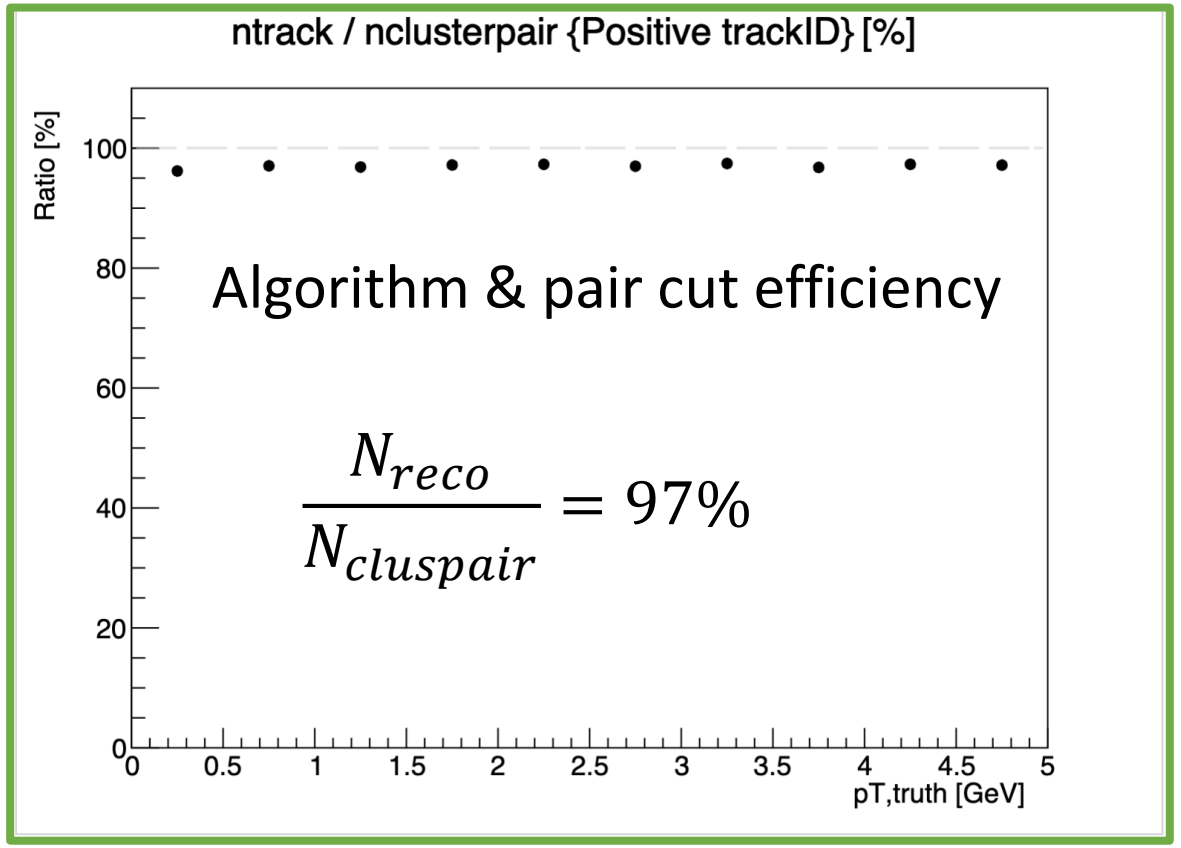
# Decomposition of the efficiency

$$eff = \frac{N_{reco}}{N_{truth}} = \frac{N_{cluspair}}{N_{truth}} \cdot \frac{N_{cluspair}}{N_{reco}}$$




$$\frac{N_{cluspair}}{N_{truth}} = 86.9\%$$

Why low?

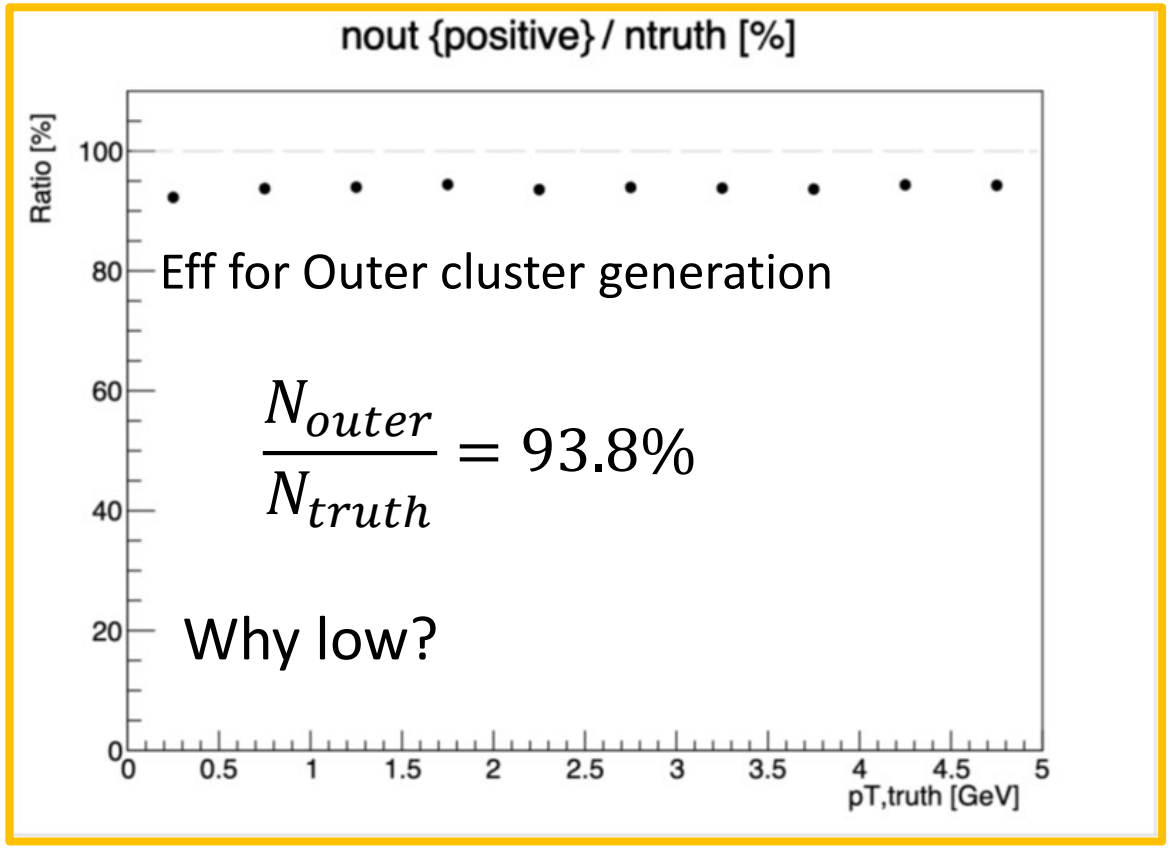



Algorithm & pair cut efficiency

$$\frac{N_{reco}}{N_{cluspair}} = 97\%$$

# Further decomposition of the efficiency

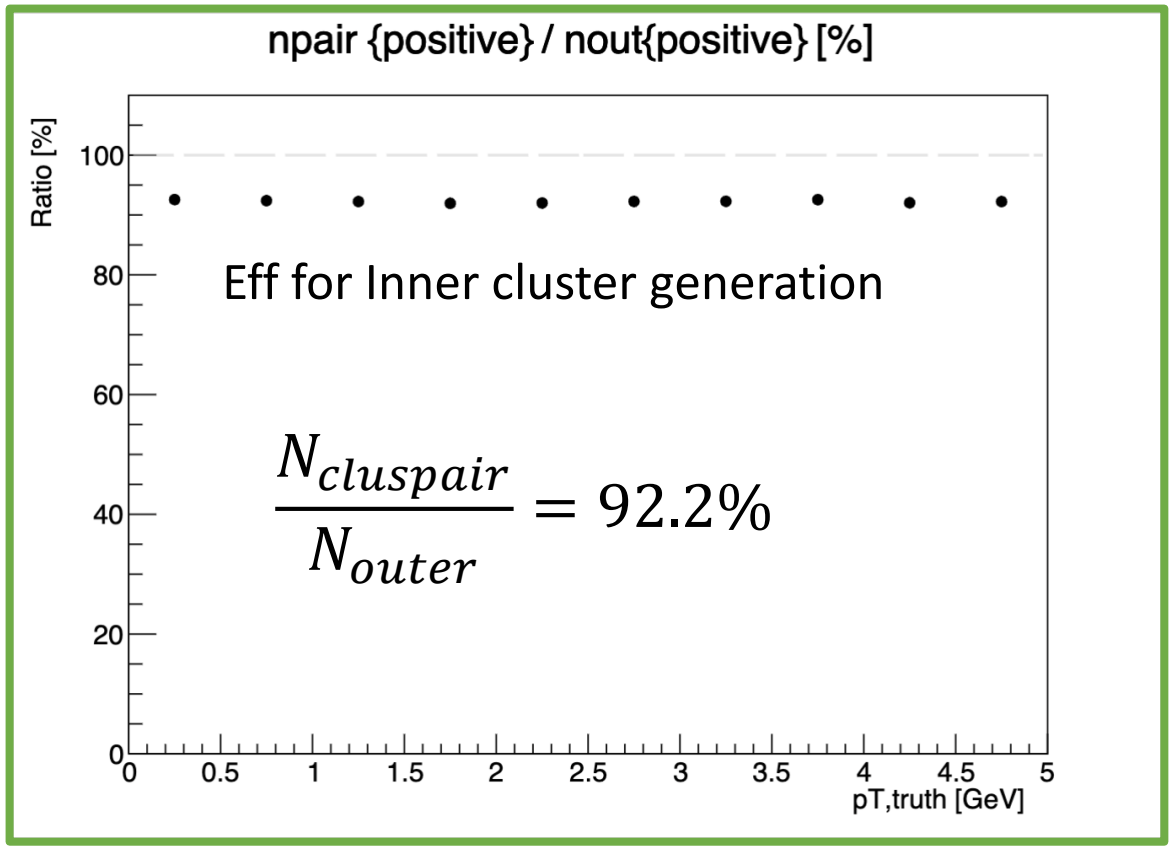
$$\frac{N_{cluspair}}{N_{truth}} = \frac{N_{outer}}{N_{truth}} \cdot \frac{N_{cluspair}}{N_{outer}} = 93.8 \times 92.2 = 86.4\%$$




Eff for Outer cluster generation

$$\frac{N_{outer}}{N_{truth}} = 93.8\%$$

Why low?

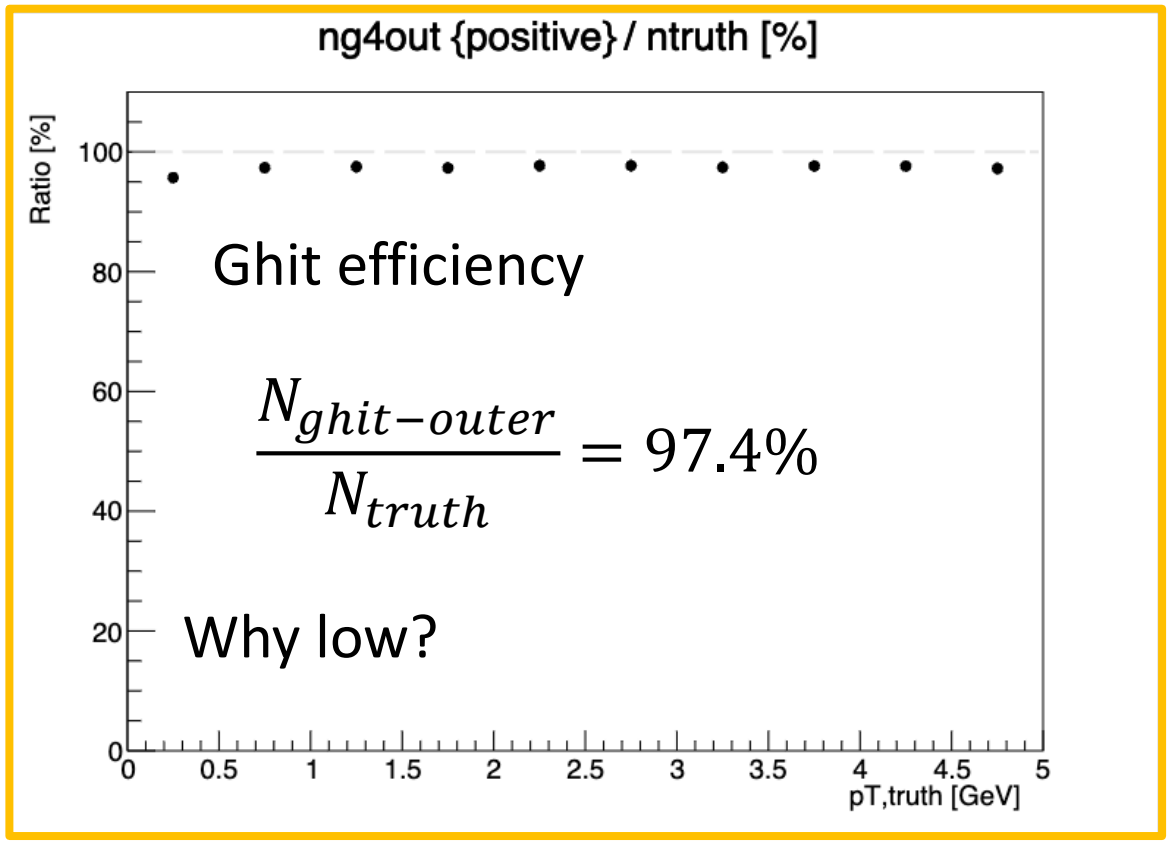



Eff for Inner cluster generation

$$\frac{N_{cluspair}}{N_{outer}} = 92.2\%$$

# Further decomposition 2

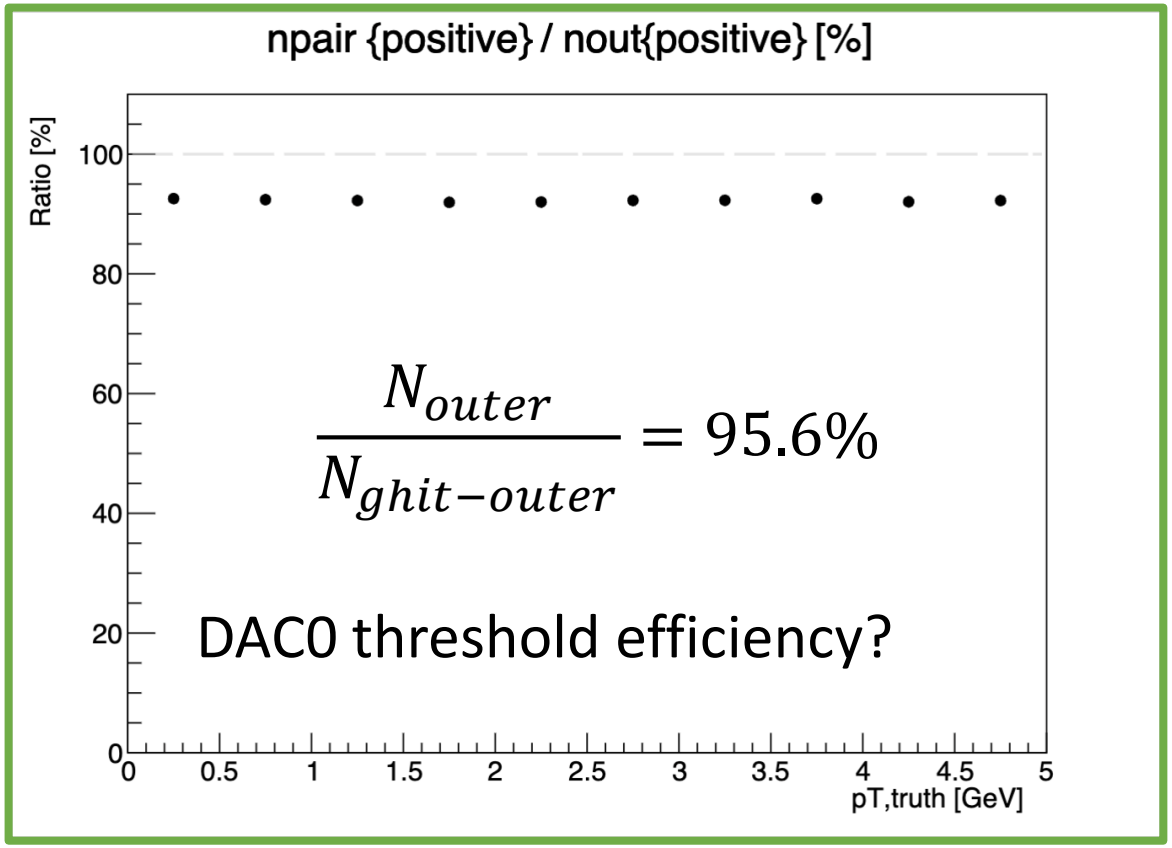
$$\frac{N_{outer}}{N_{truth}} = \frac{N_{ghit-outer}}{N_{truth}} \cdot \frac{N_{outer}}{N_{ghit-outer}} = 97.4 \times 95.6 = 93.1\%$$




Ghit efficiency

$$\frac{N_{ghit-outer}}{N_{truth}} = 97.4\%$$

Why low?




$$\frac{N_{outer}}{N_{ghit-outer}} = 95.6\%$$

DAC0 threshold efficiency?

# Reason

Ghit efficiency ： $\frac{N_{ghit-outer}}{N_{truth}} = 97.4\%$

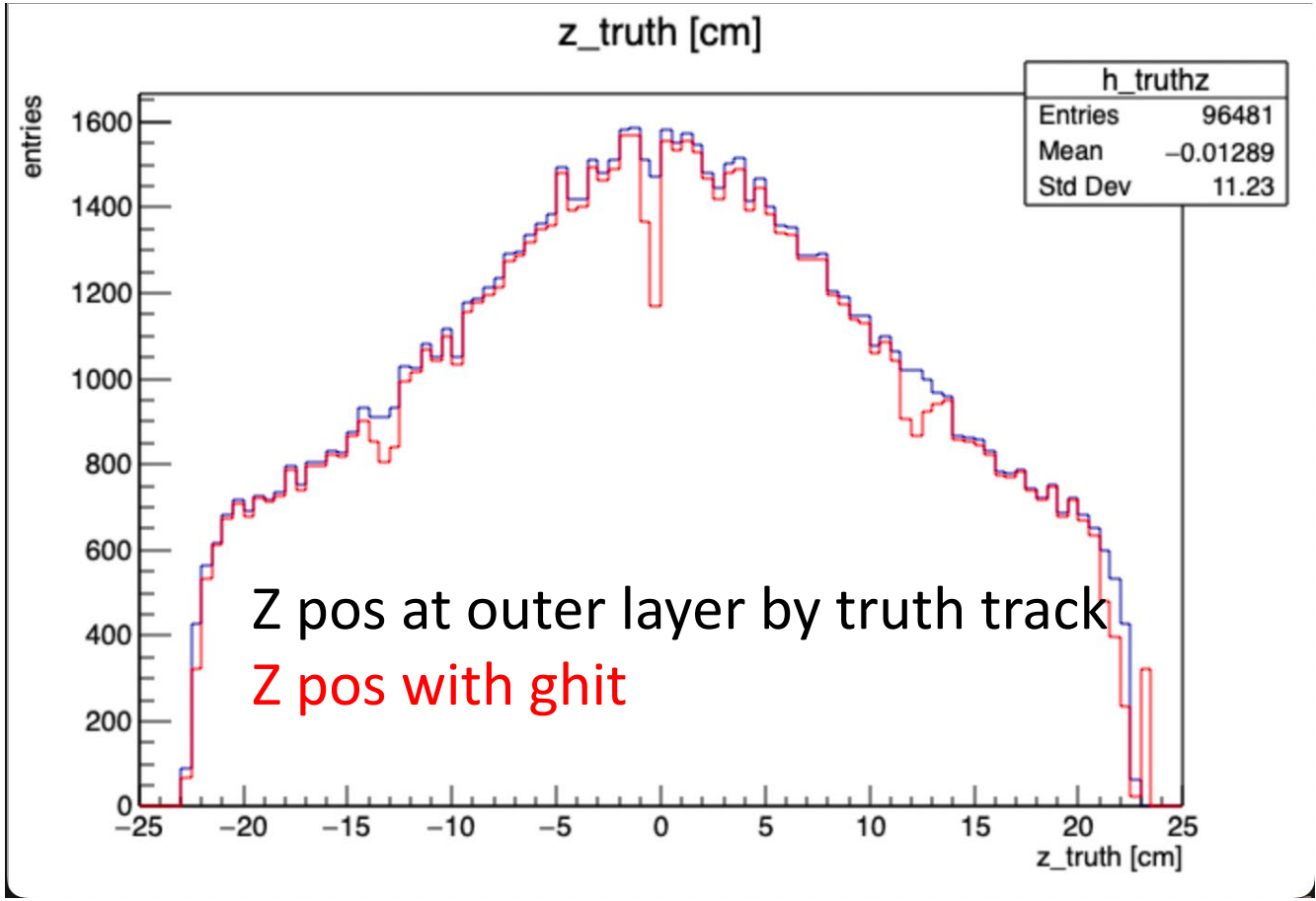
Inactive areas btw sensors cause 3% efficiency loss

INTT tracklet eff is studied

- 84% eff is described by MC

Hinako made a connection chain

Gtrack -> Ghit -> Hit (reco) -> Cluster (reco)

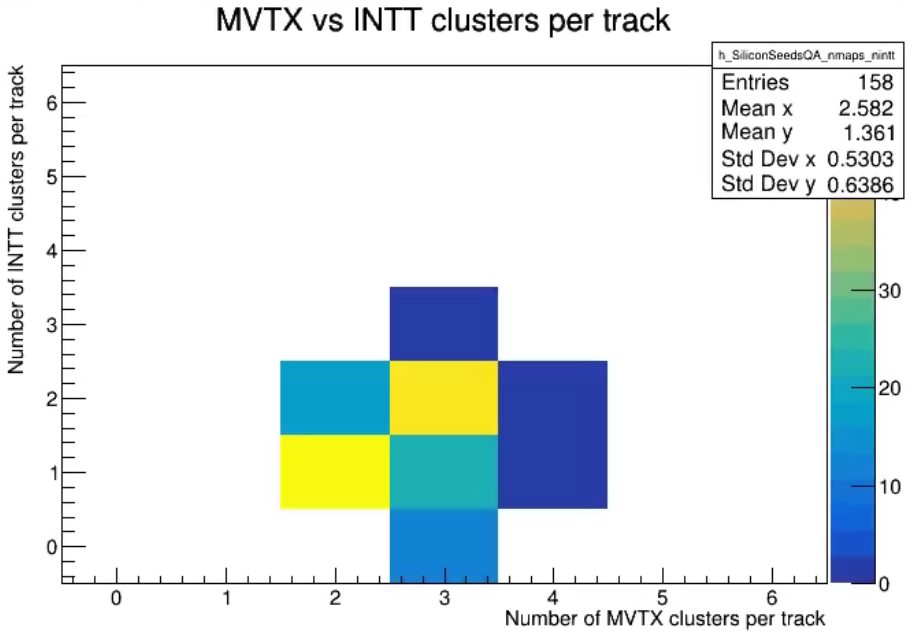


# Si seeding analysis

- Example : SiliconSeedsQA

```
auto trackmap = findNode::getClass<SvtxTrackMap>(topNode, m_trackMapName);
for (const auto &[key, track] : *trackmap)
{
     auto ckeys = get_cluster_keys(track);
     float eta = track->get_eta();
     float phi = track->get_phi();
     float pt = track->get_pt();
     int charge = track->get_charge();
     float chi2ndf = track->get_quality();

     int trkcrossing = track->get_crossing();
```

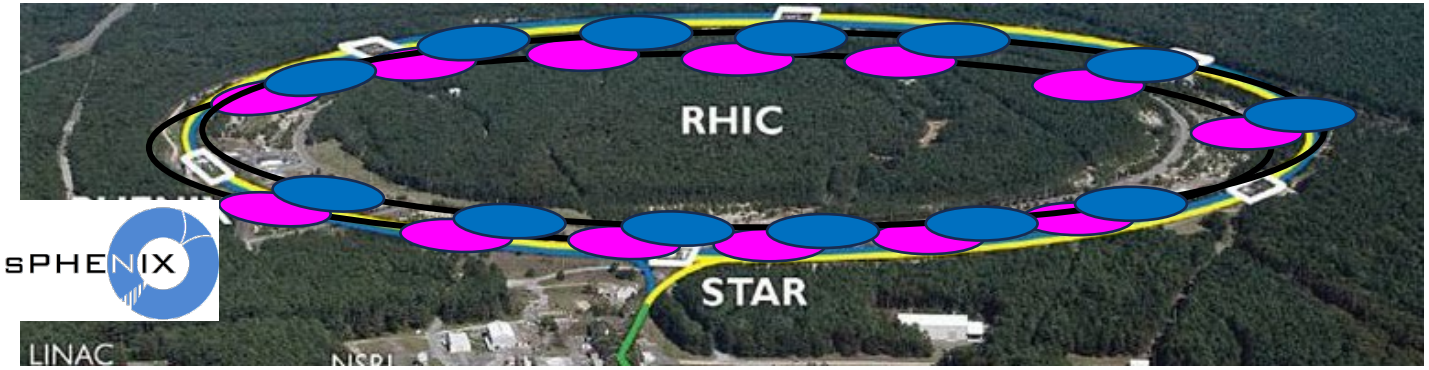


# Summary

- Basic INTT software developed and available
  - All sPHENIX codes are managed by Github
- Workflow of data processing explained
  - DST production
- Data you can use for your analyses

- Si-seeding opens opportunities for Quarkonia measurements and Jet

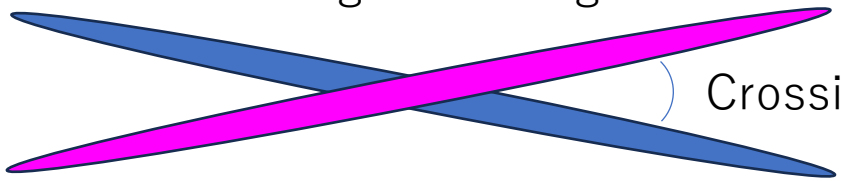
# Luminosity vs Crossing angle

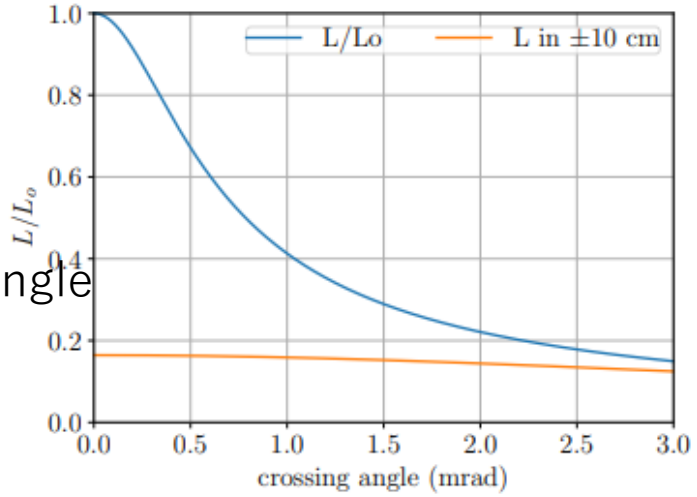



sPHENIX without xing angle – what can we give STAR without impacting sPHENIX

Zero angle crossing

Finite angle crossing




- sPHENIX crossing angle implemented for beam-beam suppression and to maximize collisions within ±10 cm.
- 4.5x increase in luminosity if going to head-on and looking at the full luminosity distribution
- IR8 D0 polarity will need to be switched back to nominal for head-on collisions (4 hours on maintenance day)