Seed Finding in ACTS - CPU & GPU

Luis Falda Coelho[1]

[1]CERN + University of Coimbra + LIP

This presentation contains contributions of many people within ACTS group

ACTS Developers Workshop

9 Nov 2023

Seeding in ACTS

ACTS - Experiment-independent toolkit for charged particle track reconstruction in HEP [arXiv:2106.13593]

Seed finding is an important and computationally expensive problem in reconstruction

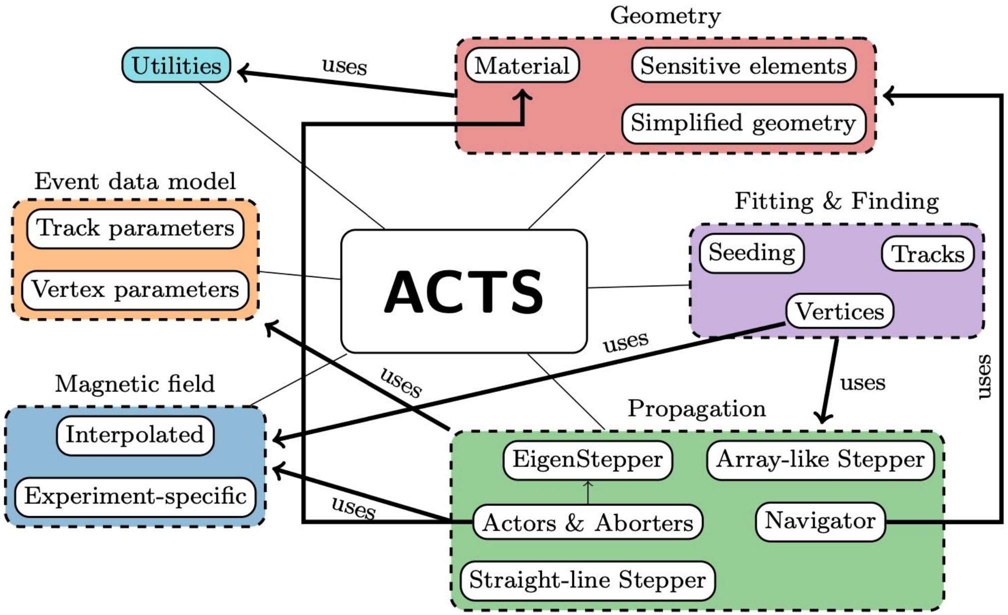
ACTS provides highly optimized seed finding algorithms

CPU-based seeding strategies used by a broad range of detectors:

- Default Seeding (Mid-point seeding)
- Orthogonal Seeding
- Truth-based

R&D platform to explore new techniques:

- GPU parallelisation of seeding
- ML-based seed filtering (Coretin's talk)



CPU Based Seeding - ACTS Default Seeding

CPU Based Seeding - ACTS Default Seeding

The Mid-Point Seeding Algorithm

Essential part of the tracking chain:

- Track finding uses seeds parameters as first estimate of track direction and momentum

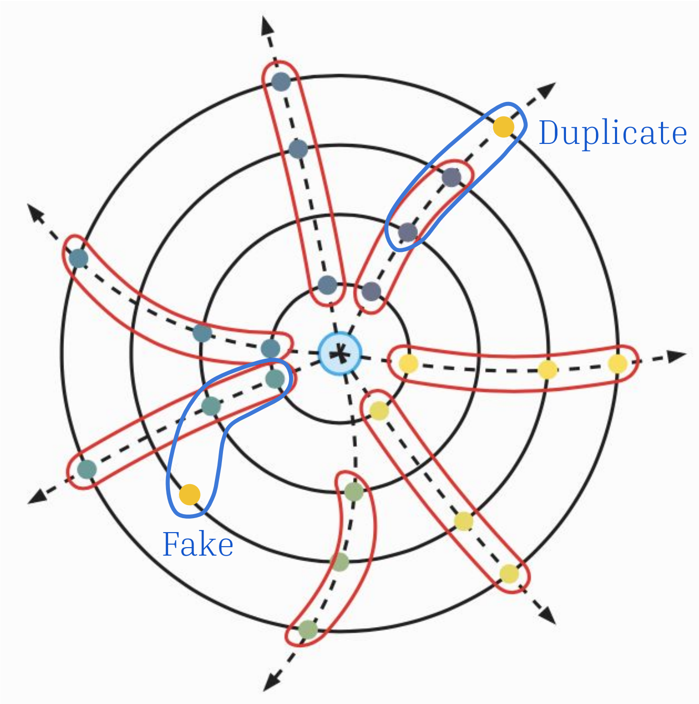
Start by combining 3D representation of detector measurements (SPs)

Finding too many duplicated or fake seeds increase the time needed for tracking

SeedFinder.ipp → **SeedFilter.ipp**

- SeedFinder.ipp: Build SP combinations; Filter unwanted combinations
- SeedFilter.ipp: Keep only the best combinations (Seed confirmation)

Large number of parameters (some are specific or dependent on geometry) → ACTS can automatically tune the parameter based on tracking performance (Rocky's talk from CTD 2022)



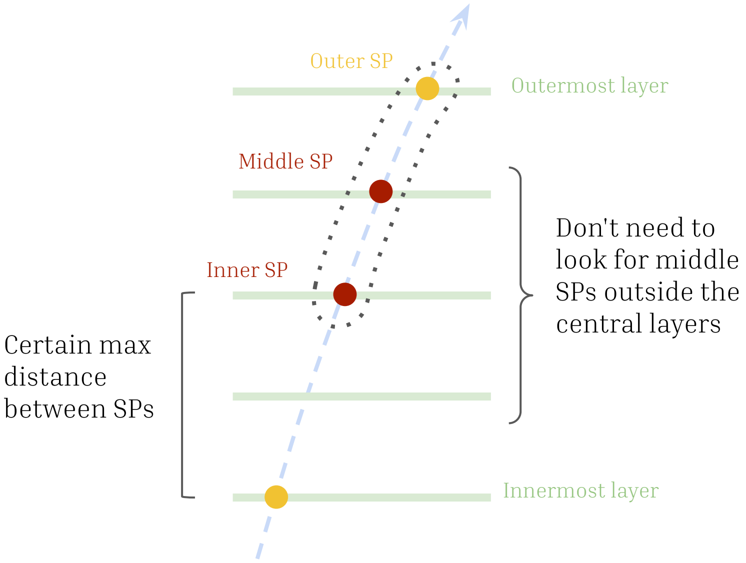
ACTS Seeding Documentation

CPU Based Seeding - ACTS Default Seeding

Seed Finding (SeedFinder.ipp):

Track seeds are created by combining SPs

- Built assuming helical path from the center of detector in a homogeneous magnetic field
- Starts from selecting a middle SP in a certain detector layer:
    - Check if middle SP in region of interest (r and z)



- Duplets formed from inner and outer SPs:
    - Check compatibility between duplets (Δr, Δz, z_0 within collision region, forward angle within bounds)
- Accepted inner and outer SPs sharing same middle SP to build triplets:
    - r-z slope compatibility with maximum multiple scattering effect (produced by the minimum allowed p_T particle) + a certain uncertainty
    - Helix radius required to be greater than the minimum allowed radius
    - Transverse impact parameter d_0 to be within bounds

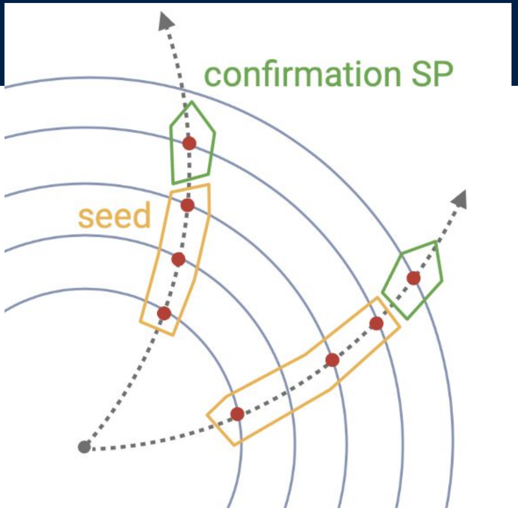
$$\left(\frac{1}{\tan\theta_b} - \frac{1}{\tan\theta_t}\right)^2 < \sigma^2_{p_T^{min}} + \sigma^2_f,$$

CPU Based Seeding - ACTS Default Seeding

Seed Finding (SeedFinder.ipp):

Seed Finder Configuration	
`rMax rMin zMin zMax phiMin phiMax`	Definition of region of interest in (r,z,ϕ) for all SPs
`rMinMiddle rMaxMiddle`	Minimum and maximum r boundaries for middle SPs
`deltaRMinTopSP deltaRMaxTopSP`	Minimum and maximum radial distance between middle-outer doublet components
`deltaRMinBottomSP deltaRMaxBottomSP`	Minimum and maximum radial distance between inner-middle doublet components
`deltaZMax`	Maximum value of z-distance between SPs in doublet
`cotThetaMax`	Maximum allowed $\cot(\theta)$ between two SPs in doublet
`collisionRegionMin collisionRegionMax`	Limiting location of collision region in z-axis used to check if doublet origin is within reasonable bounds
`minPt`	Minimum allowed value for the transverse momentum of particles
`sigmaScattering`	Number of sigmas of scattering angle to be considered in the minimum p_T scattering term
`radLengthPerSeed`	Term that accounts for the thickness of scattering medium in radiation lengths in the Lynch & Dahl correction to the Highland equation [2, 3]
`maxPtScattering`	Maximum transverse momentum for scattering calculation
`impactMax`	Maximum value of impact parameter estimation of the seed candidate

CPU Based Seeding - ACTS Default Seeding



Seed Confirmation (SeedFilter.ipp):

After selecting the SP-triplets, a seed confirmation procedure is applied to all the triplet combinations:

- Compare seeds with similar helix radius
- Rank the seeds based on a customisable weight

$$w = (c_1 \cdot N_t - c_2 \cdot d_0 - c_3|z_0|) + \text{detector specific cuts}$$

More measurements leads to higher quality

Smaller IP → higher probability of track arriving from the interaction point

Seed Confirmation Configuration	
`deltaInvHelixDiameter`	Allowed difference in curvature between two compatible seeds
`deltaRMin`	Minimum distance between compatible top SPs to be considered
`compatSeedWeight`	c_1 factor in Equation 1 for seed score calculation
`impactWeightFactor`	c_2 factor in Equation 1 for seed score calculation
`zOriginWeightFactor`	c_3 factor in Equation 1 for seed score calculation
`maxSeedsPerSpM`	Maximum number minus one of accepted seeds per middle SP

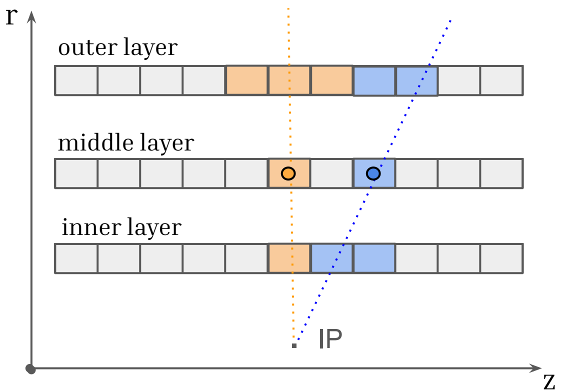
Improving the quality of the final track collections by rejecting lower-quality seeds

In the end we keep only the best ranked seeds

CPU Based Seeding - ACTS Default Seeding

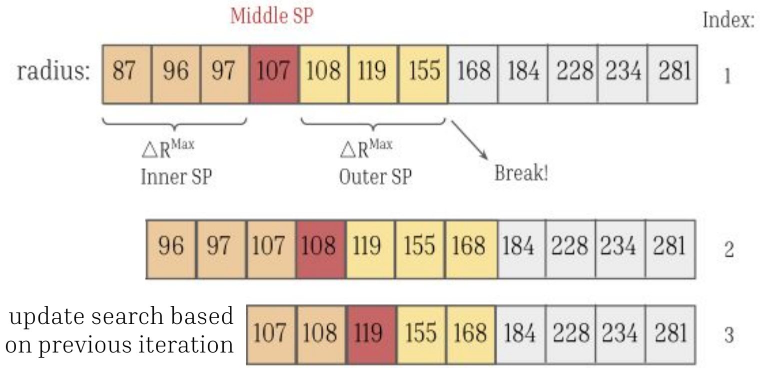
Dealing With High Multiplicity:

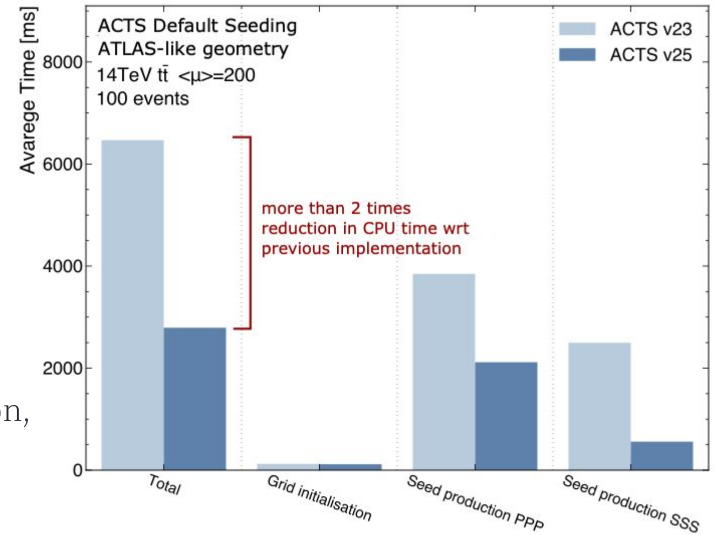
Geometrical assumptions to limit the search to certain neighbouring cells (strongly inspired by ATLAS Phase-II software developments and physics performance studies)



Sort the objects based on distinct criteria (SP radius, doublet slope, triplet helix radius):

- Binary search to avoid looking outside the region of interest:





Adapting and adding selection cuts, improving the logic, memory allocation, internal classes, hot spots + other features

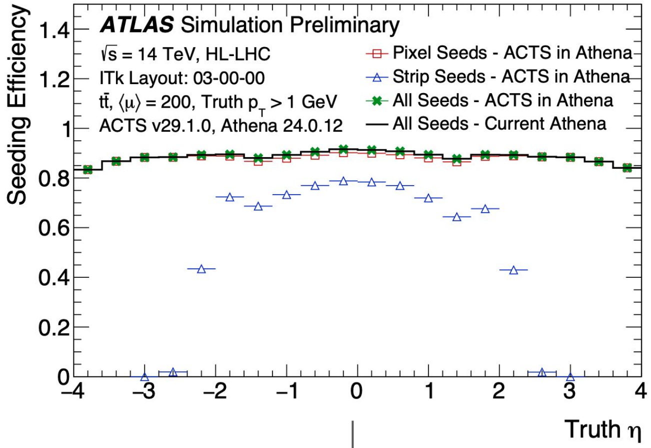
ACTS Default Seeding Flexibility

The seeding algorithm that is now implemented focus on LHC-like experiments:

- Requires B field:
  - Parameter estimation crashes w/ B=0
  - Variables used in the helix cut will result in infinity (eg. `minHelixDiameter2` and `sigmapT2perRadius`) - cut needs to be bypassed
  - Some experiments have no magnetic field or not along z axis
- Optimised for prompt particles (primary particles from the IP):
  - Some cuts assume the seed has origin compatible with IP, and without them, the execution time will explode

Trigger Discussion:

- Additional developments to include seeding for other signatures with main concern on CPU
- Development of additional seeding algorithm that can optimally work with fixed target experiment and/or in absence of magnetic field



Can provide identical physics performance as ATLAS ITk Tracking chain (Paul's talk at CTD 2023)

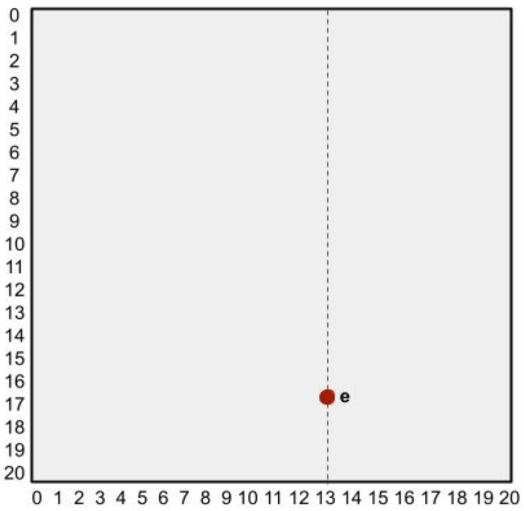
CPU Based Seeding - ACTS Orthogonal Seeding

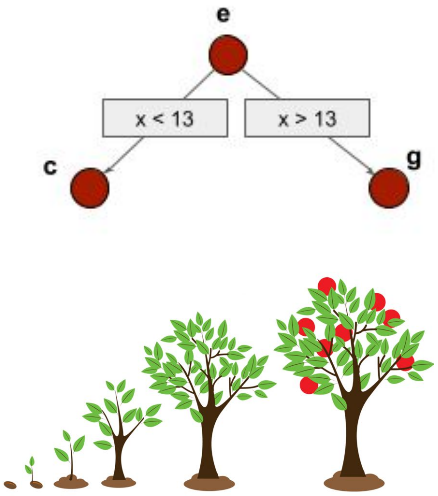
CPU Based Seeding - ACTS Orthogonal Seeding

Reverse the Logic of Traditional Seeding!

First define **z-r-φ** volumes to search for (using KD-trees) and then extract the SPs within that volume

KD-tree splits the data into smaller subspaces





Splitting is performed along one of the **z-r-φ** dimensions of the SPs, based on median value of that dimension:

- Sorting the range to find the median is too expensive O(n log n). But we can use the middle value between the min and max O(1)
- Dynamic median finding: exact median for small data sets, approximate median for larger data sets
- Balanced KD-tree can improve search performance

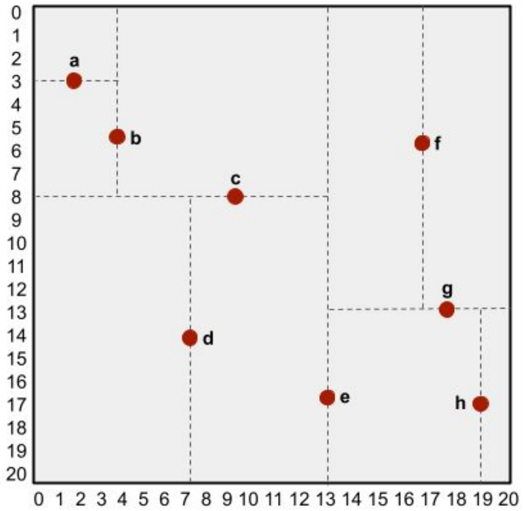
This creates two child nodes containing points that are smaller or larger than the median value

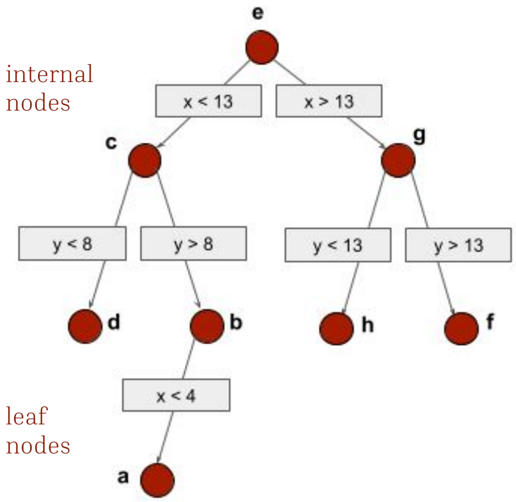
Stephen's talk on ACTS workshop 2022

Reverse the Logic of Traditional Seeding!

First define **z-r-φ** volumes to search for (using KD-trees) and then extract the SPs within that volume

KD-tree splits the data into smaller subspaces





Splitting is recursively applied to each child node, alternating the dimension along which the split is performed:

- Results in a binary tree structure
- Each node represents a subspace of the data

Range search on the tree for each middle SP:

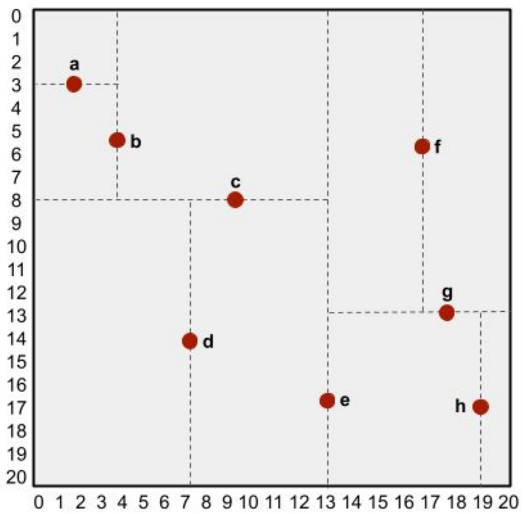
- First we select the search range for that SP based on similar constraints as default seeding
- But now we need to "orthogonalize" this cuts on the tree dimensions

CPU Based Seeding - ACTS Orthogonal Seeding

Reverse the Logic of Traditional Seeding!

First define **z-r-φ** volumes to search for (using KD-trees) and then extract the SPs within that volume

KD-tree splits the data into smaller subspaces



"Orthogonalizing" means making search range cuts perpendicular to the tree dimensions:

- Trivial for some cases:

$$g_{\min}(\vec{x}) = x_\phi - \Delta\phi_{\max}$$
$$g_{\max}(\vec{x}) = x_\phi + \Delta\phi_{\max}$$

φ should not change too much between SPs

- But not for all of them:

$$z_0^{min} < z_M - r_M \times \frac{\Delta Z}{\Delta R} < z_0^{max}$$

duplet would have originated from the collision point

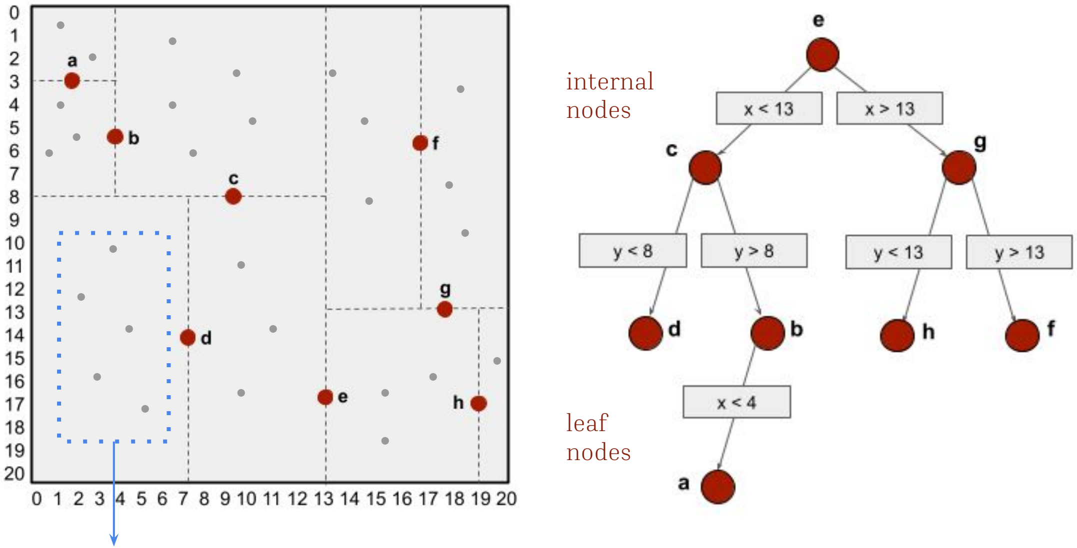
  - Define a weaker version in orthogonal fashion
  - Can always check the tighter constraint in a non-orthogonal way later

CPU Based Seeding - ACTS Orthogonal Seeding

Reverse the Logic of Traditional Seeding!

First define **z-r-φ** volumes to search for (using KD-trees) and then extract the SPs within that volume

KD-tree splits the data into smaller subspaces



Bounding Box - It is the smallest box that contains all the points within that node

Performing a query on the tree:

- Algorithm traverses the tree by comparing a certain middle SP to the splitting value of each node
- In case the node's bounding box is fully contained within the region of interest we build compatible SP combinations
- Not necessarily a leaf node - stop the search if reached an internal node that is fully contained by the search range
- If leaf node is not fully contained - we take only the SPs that are

CPU Based Seeding - ACTS Orthogonal Seeding

Less parameters than the default seeding since we do not need a grid

Three specific parameters from orthogonal seeding:

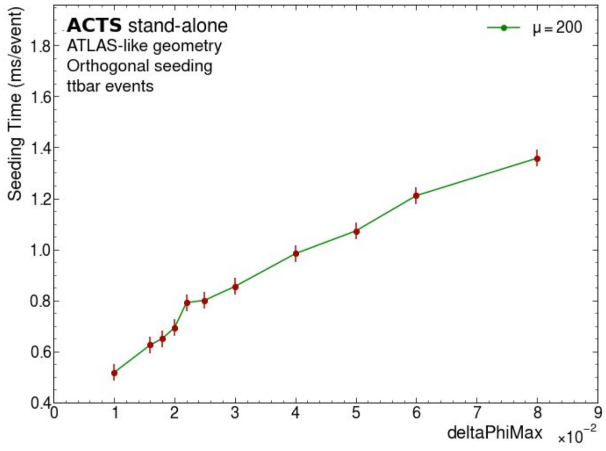
max_exact_median - determines the maximum number of elements where we still calculate the exact median

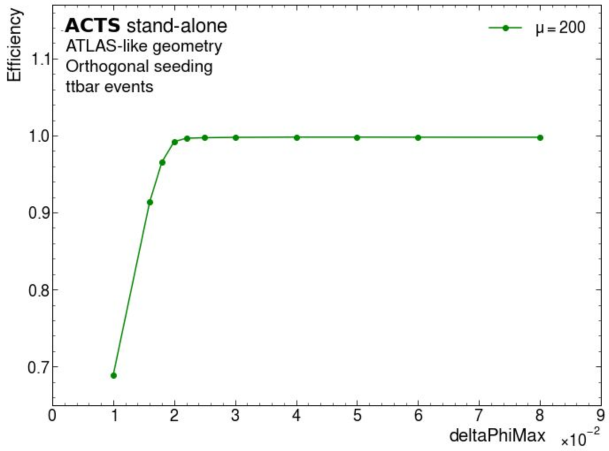
LeafSize - The maximum number of elements stored in a leaf node

→ Not configurable and do not seem to affect much the performance for high pile-up

deltaPhiMax - Shrink the φ range of middle SP, analogous to φ bin size in grid from default seeding

→ Can significantly affect the performance and should be optimized





CPU Based Seeding - ACTS Orthogonal Seeding

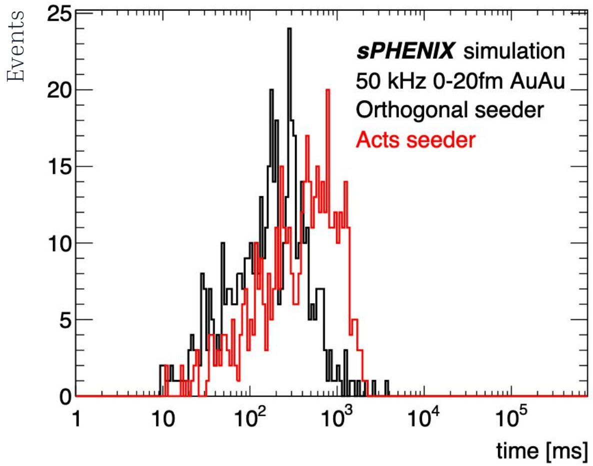
Can be expanded to map points into 6D (**x,y,z,φ,θ,r**) space! More discriminatory power

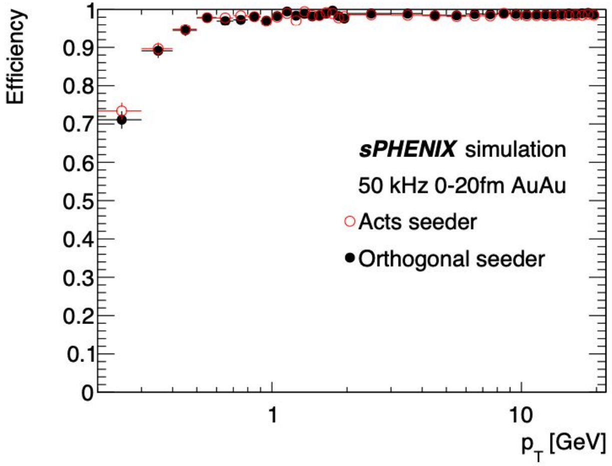
KD-tree could be extended using time as a fourth dimension to support future timing detectors

KD-trees are considered GPU-friendly data structures, which can be efficiently constructed and queried in a massively parallel environment → could be powerful in GPUs!

Good physics performance but depends a lot on the detector layout and the seeding configuration:

- Must find highly restricted search spaces in order to outperform the default seeding





R&D Lines for Seeding - GPU based Seeding

R&D Lines for Seeding with ACTS

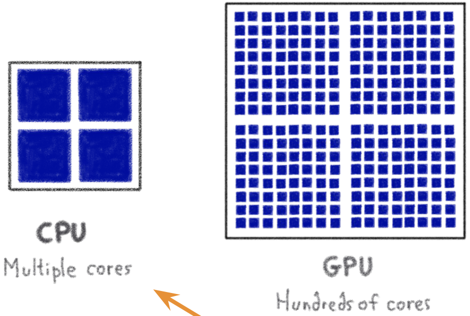
Two dedicated R&D lines in ACTS:

- Parallel code execution, mainly focus on GPU accelerators and portability
- Machine learning based/inspired modules (Coretin's talk)

GPU Based Seeding:

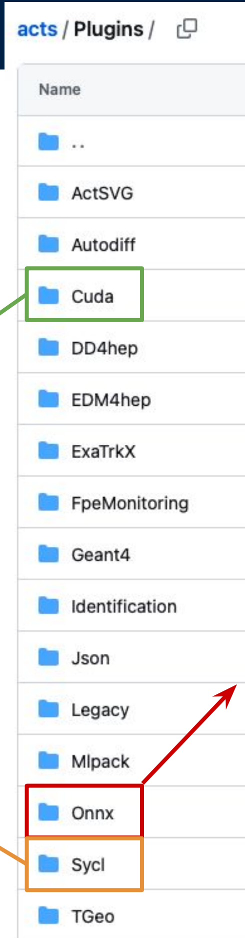
R&D to offload tracking algorithms to GPUs

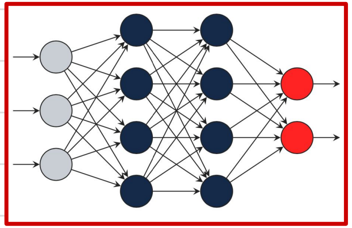
GPU-based tracking tools with ACTS are available (Seeding, Kalman Filter, geometry navigator)



Requirements:

- Same physics performance as the existing CPU algorithms
- Realistic detector setup
- Event Data Model (EDM) shared by CPU and GPU
- Primarily focusing on CUDA and SYCL implementations





Replace tracking components by ML

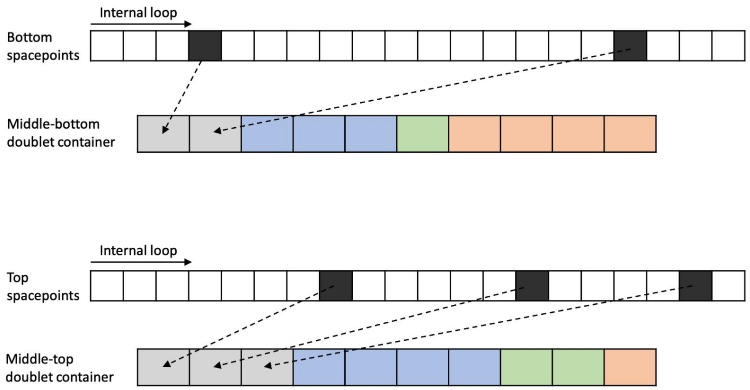
GPU Based Seeding

Similar algorithm as CPU-based seeding:

- But need to count the number of item objects (SPs or doublets or triplets) for each stage of the seeding to properly allocated the workload between the different cores and pre-assign the memory space for the objects

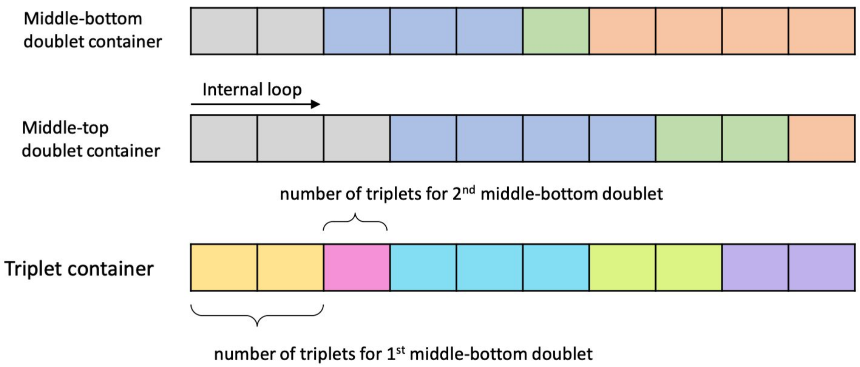
Multiple thread blocks for a certain (φ, z) grid bin of the detector

Dublet Finding:



Every thread (for a compatible middle SP) iterates over inner and outer SPs in neighbour bins to record the doublet objects

Triplet Finding:



Every thread (for a compatible middle-bottom doublet) iterates over middle-top doublets, whose middle SP is the same, to record the triplet objects

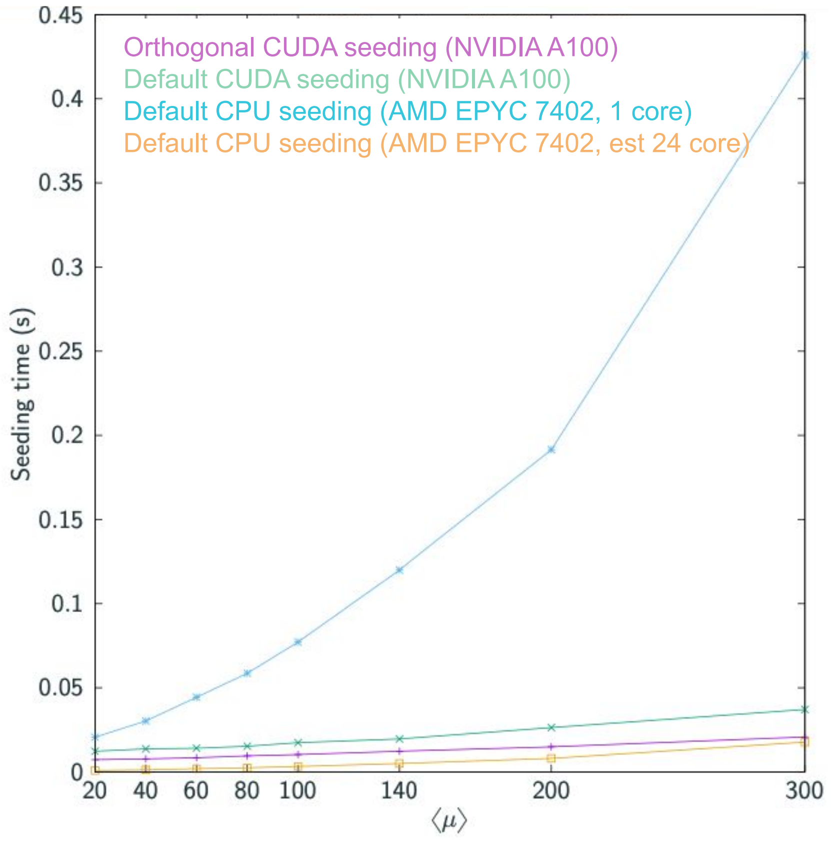
Seed Filter + Confirmation: similar approach

GPU Based Seeding

ACTS parallelization research and development project (traccc) implements both seeding algorithms for GPGPU devices

- Implemented in CUDA - the primary programming platform for NVIDIA GPGPUs
- and SYCL - a vendor-agnostic programming model developed by the Khronos Group

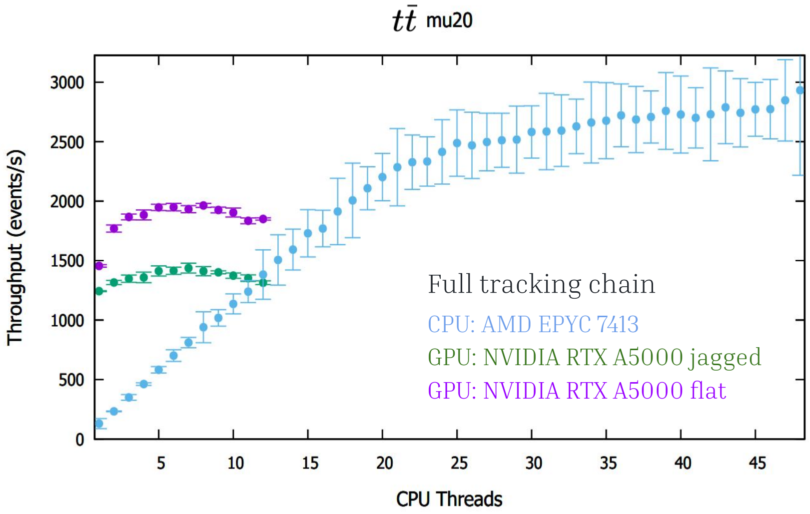
Seeding with GPU is worthwhile only above a certain level of pileup

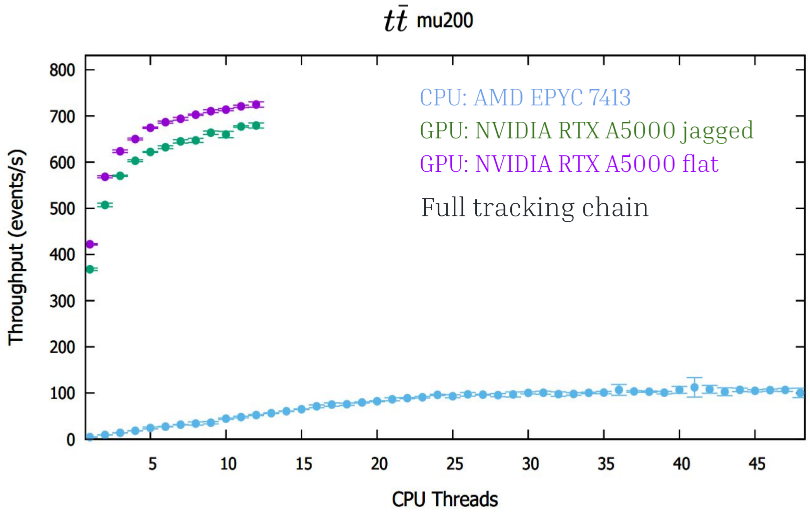


GPU Based Seeding

Seeding with a flat(ter) EDM:

- Seeding requires dealing with nested data structures where inner vectors can have different lengths for storage of doublets and triplets
- Testing the effect of removing jagged vectors by allocating a single large flat vector

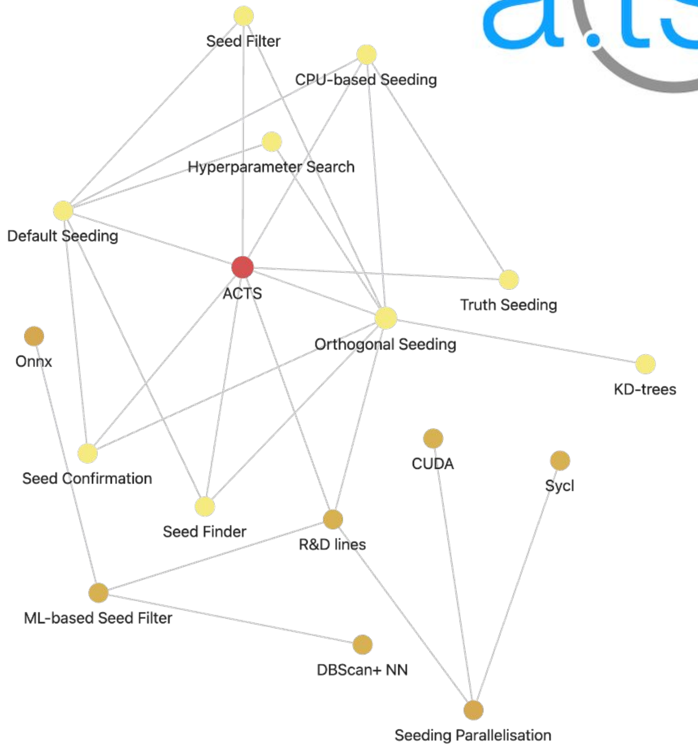




Summary

ACTS incorporates CPU and GPU-based seeding algorithms:

- A lot of work is necessary to customise the seeding to achieve optimal performance
- ACTS seeding has shown to be quite customisable and perform well across numerous experiments
- May not be optimal in some cases (eg. absence of magnetic field, LLP)
- Consistent effort towards enhancing flexibility and performance
- Optimizing code for parallel processing, with a primary emphasis on GPU acceleration and portability
- Yielding promising results



Backup

Approximating the z-intercept

Weakened version can be found by reasoning backwards: instead of starting with two points and checking whether they intersect within a z boundary, start with the z boundary and one point, check where the second point may be

$$g_{\min}(\vec{x}) = z_{\max} - \frac{r_{\max}(z_{\max} - x_z)}{x_r}$$

$$g_{\max}(\vec{x}) = z_{\min} + \frac{r_{\max}(x_z - z_{\min})}{x_r}$$