Towards a GPU-enabled electron seeding algorithm in the CMS experiment

CHEP2024: 27th International Conference on Computing in High Energy & Nuclear Physics, Krakow, Poland

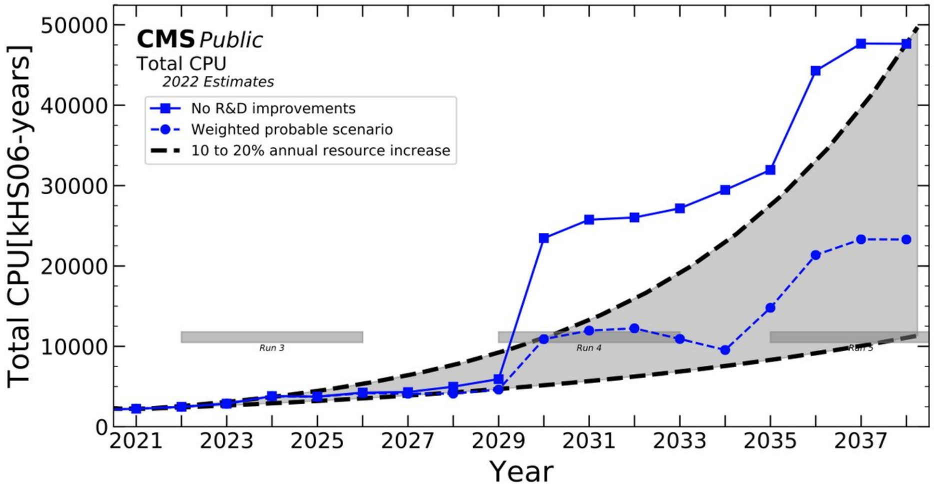
Charis-Kleio Koraka

on behalf of the CMS collaboration

October 19th - 25th 2023

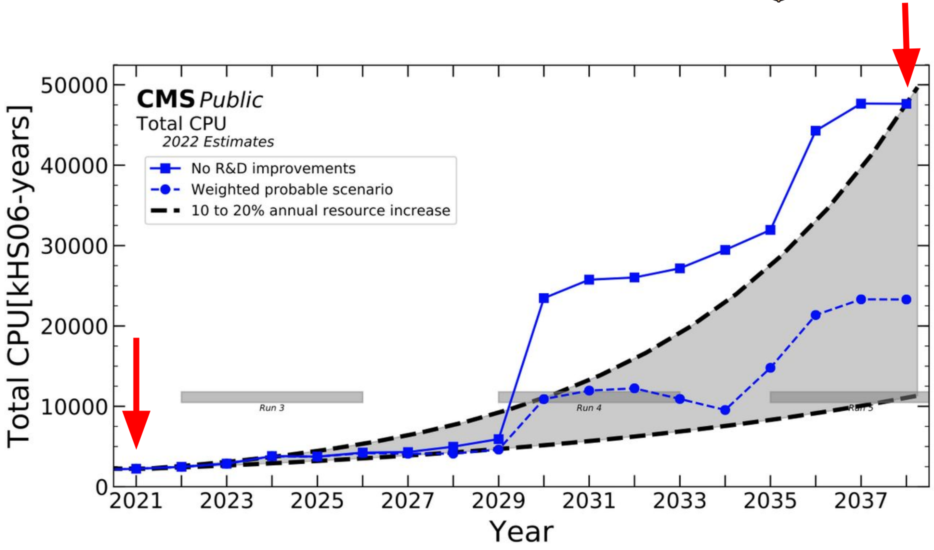
Why shift to GPUs and heterogeneous software?

- **High Luminosity LHC (HL-LHC)** will pose significant computing challenge
  - Increase in instantaneous luminosity and pile-up by more than a factor of 2
  - Upgraded detectors with higher granularity



Why shift to GPUs and heterogeneous software?

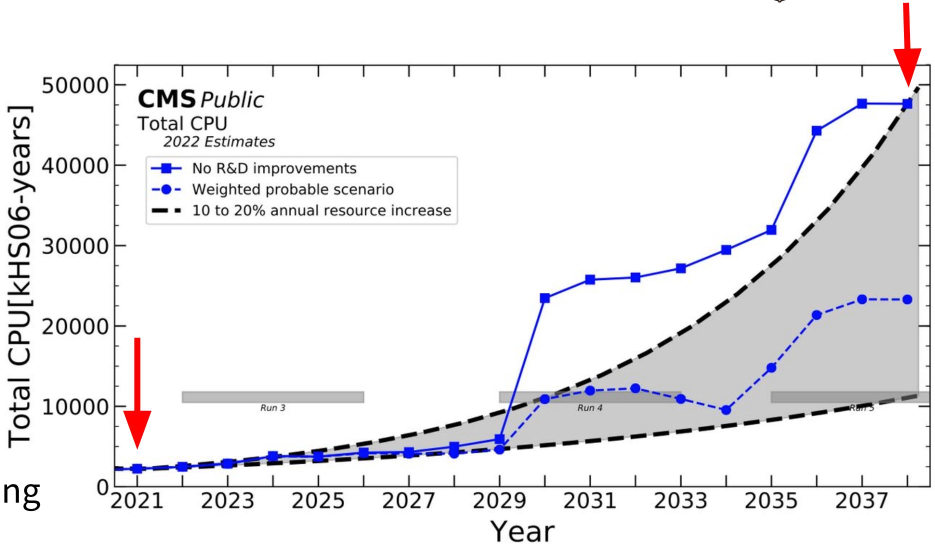
- **High Luminosity LHC (HL-LHC)** will pose significant computing challenge
  - Increase in instantaneous luminosity and pile-up by more than a factor of 2
  - Upgraded detectors with higher granularity



Might need a factor of up to 20 increase in computing resources to keep similar physics reach

Why shift to GPUs and heterogeneous software?

- **High Luminosity LHC (HL-LHC)** will pose significant computing challenge
  - Increase in instantaneous luminosity and pile-up by more than a factor of 2
  - Upgraded detectors with higher granularity

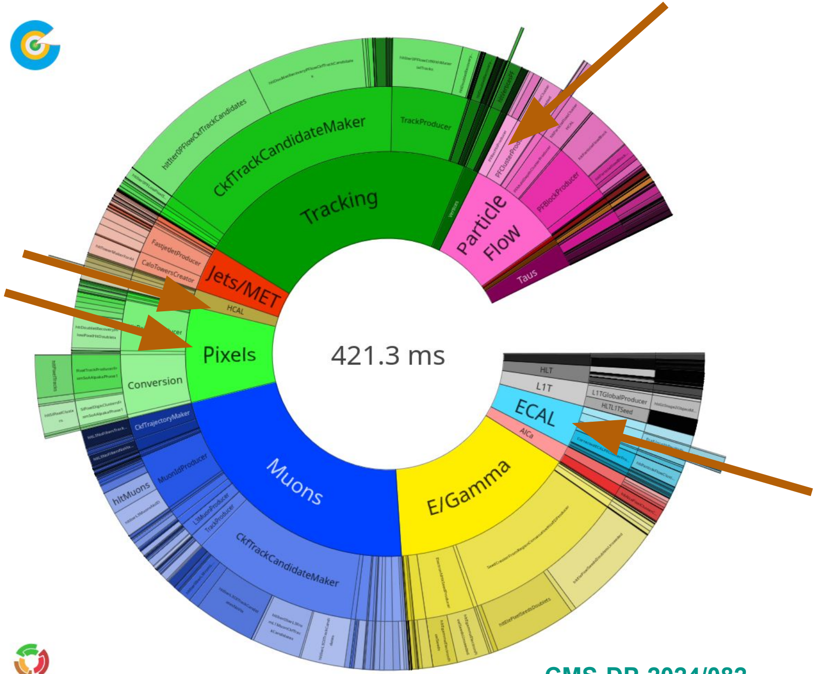


Might need a factor of up to 20 increase in computing resources to keep similar physics reach

- **GPUs and heterogeneous computing:**
  - Risk mitigating approach for computing model
  - Help cope with the higher throughput
  - Keep energy consumption low
  - Allow to utilize High Performance Computing (HPC) to address scientific challenges

GPU enabled event reconstruction@CMS

Run-3 High Level Trigger (HLT)

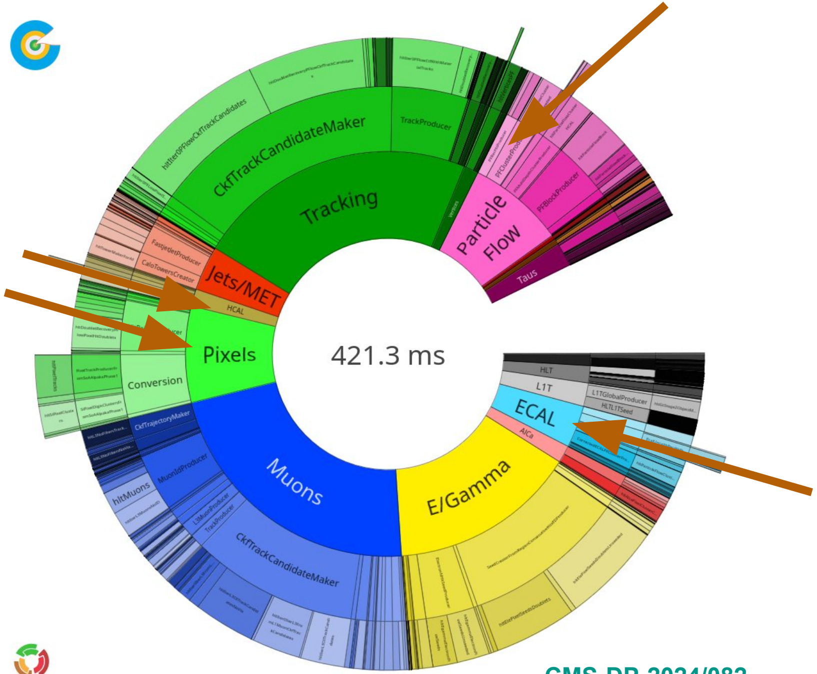


CMS-DP-2024/082

- In CMS several algorithms have already been redesigned to run on GPUs

GPU enabled event reconstruction@CMS

Run-3 High Level Trigger (HLT)



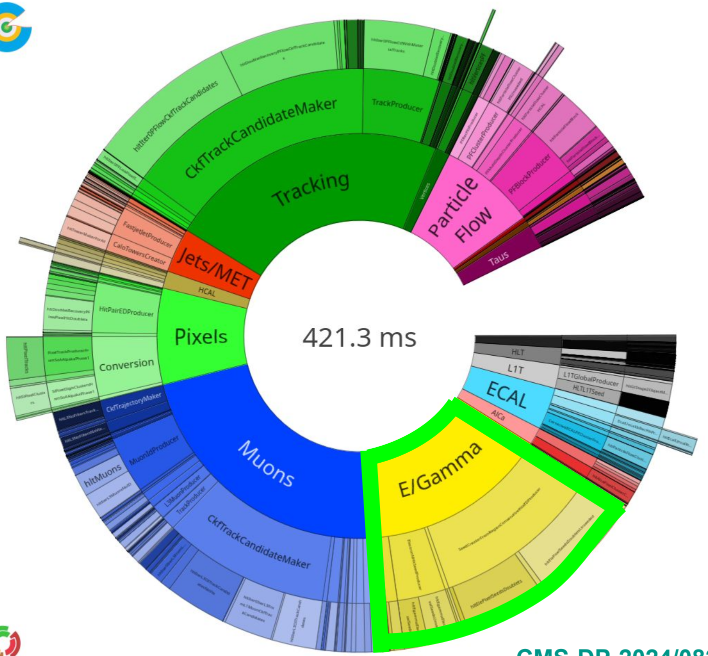
CMS-DP-2024/082

- In CMS several algorithms have already been redesigned to run on GPUs
- Since the start of Run-3 (2022) these algorithms have been used to reconstruct / collect data at HLT
- To ensure portability, the Alpaka performance portability library is used for developments

See Andreas talk for more details!

GPU enabled event reconstruction@CMS

Run-3 High Level Trigger



CMS-DP-2024/082

- Electron/photon reconstruction currently ~15% of overall reconstruction time @ HLT

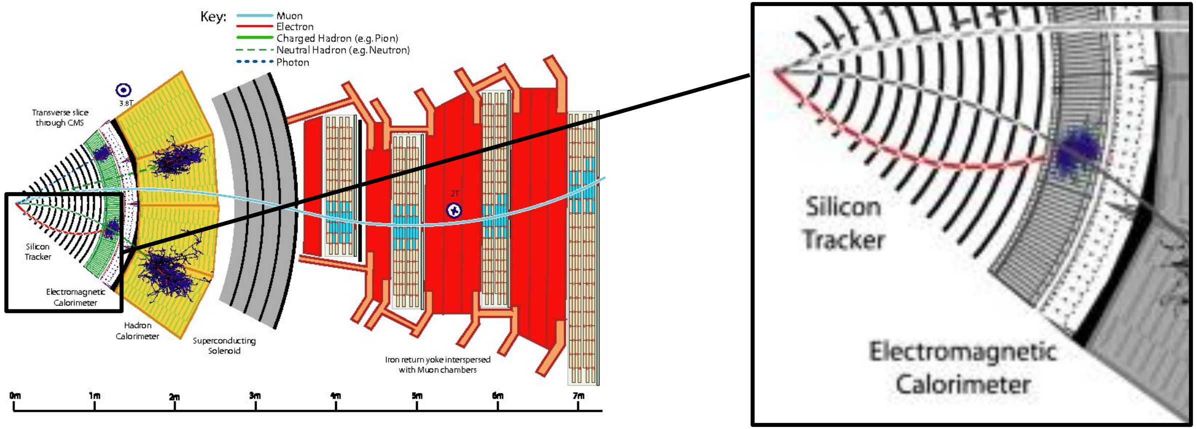
- ~90% of e/gamma reconstruction time spent on electron seeding

Electron reconstruction@CMS

A simplified description [1]

ECAL super clustering → **Electron Seeding** → GSF Tracking →

- ID variable construction
- Supercluster Refinement
- Calibration

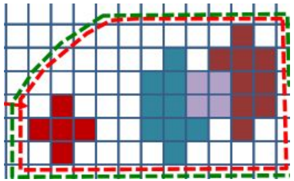


Electron reconstruction@CMS

A simplified description [1]

ECAL super clustering → Electron Seeding → GSF Tracking → ID variable construction / Supercluster Refinement / Calibration

- Combine multiple ECAL energy deposits "clusters" into a large super-cluster to capture the energy of the original electron

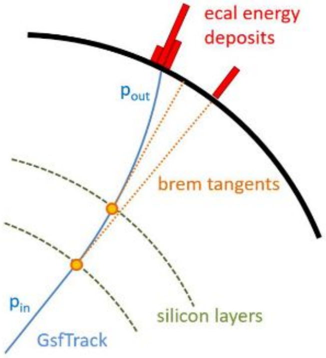


Electron reconstruction@CMS

A simplified description [1]

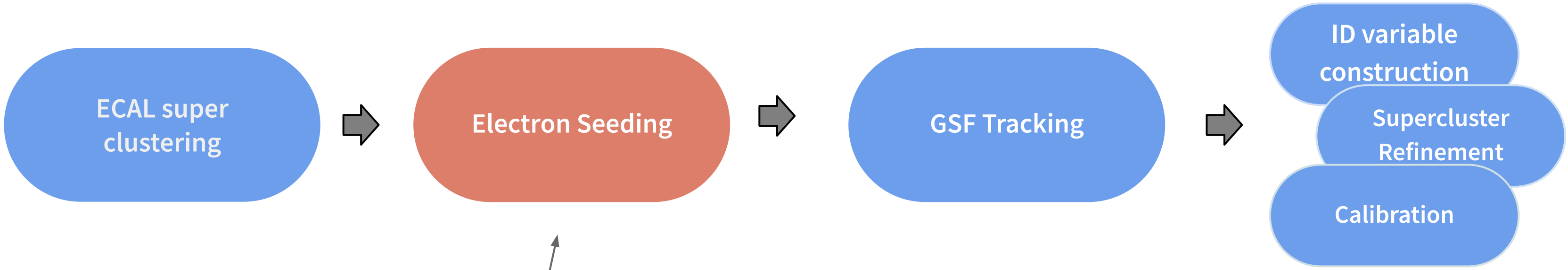
ECAL super clustering → Electron Seeding → GSF Tracking → ID variable construction / Supercluster Refinement / Calibration

- Gaussian Sum Filter track fitting algorithm



Electron reconstruction@CMS

A simplified description [1]



- **Electron seeding**
  - Identification of hit patterns that lie on an electron trajectory
  - GSF tracking CPU intensive & cannot run over all reconstructed tracker hits

Changes towards a GPU enabled seeding algorithm

Data Structures

Algorithm logic

GPU compatible utility functions

Changes towards a GPU enabled seeding algorithm

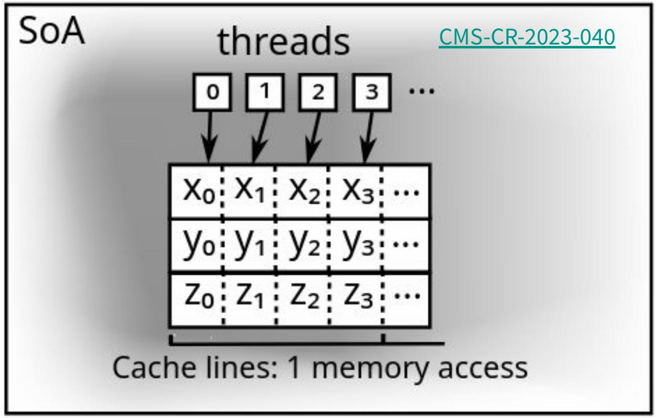
Data Structures | Algorithm logic | GPU compatible utility functions

- **Two data collections needed for the electron seeding step :**
  - Collection of ECAL super clusters (~O(10))
  - Collection of tracker seeds (combination of 2 or 3 pixel tracker hits) (~$O(10^4)$)

Changes towards a GPU enabled seeding algorithm

Data Structures | Algorithm logic | GPU compatible utility functions

- **Two data collections needed for the electron seeding step :**
  - Collection of ECAL super clusters (~O(10))
  - Collection of tracker seeds (combination of 2 or 3 pixel tracker hits) (~O(10^4))
- **Made use of CMS generic SoA data structure**
  - Designed to ensure good memory access performance



Changes towards a GPU enabled seeding algorithm

Data Structures | Algorithm logic | GPU compatible utility functions

- Two data collections needed for the electron seedi
  -
  -
- Made
  -

```
// SoA layout for supercluster
GENERATE_SOA_LAYOUT(SuperClusterSoALayout,
        // columns: one value per element
        SOA_COLUMN(double, scSeedTheta),
        SOA_COLUMN(double, scPhi),
        SOA_COLUMN(double, scR),
        SOA_COLUMN(double, scEnergy),
        SOA_COLUMN(int32_t, id)
)
```

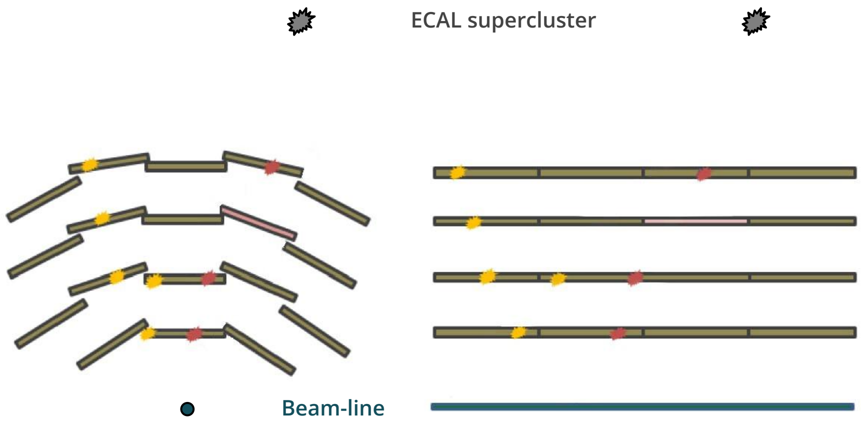
```
GENERATE_SOA_LAYOUT(EleSeedLayout,
        SOA_COLUMN(int32_t, nHits),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, hitPosX),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, hitPosY),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, hitPosZ),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, surfPosX),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, surfPosY),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, surfPosZ),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, surfRotX),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, surfRotY),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, surfRotZ),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, detectorID),
        SOA_EIGEN_COLUMN(Eigen::Vector3d, isValid),
        SOA_COLUMN(int32_t, isMatched),
        SOA_COLUMN(int32_t, matchedScID)
)
```

Changes towards a GPU enabled seeding algorithm

Data Structures

Algorithm logic

GPU compatible utility functions

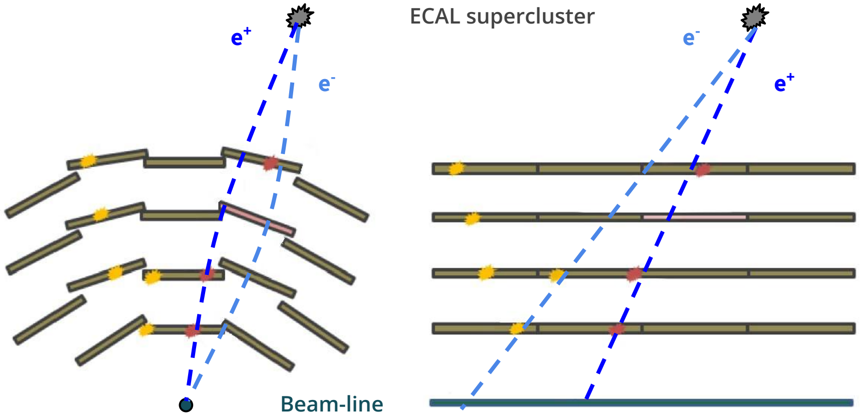


Changes towards a GPU enabled seeding algorithm

Data Structures

Algorithm logic

GPU compatible utility functions



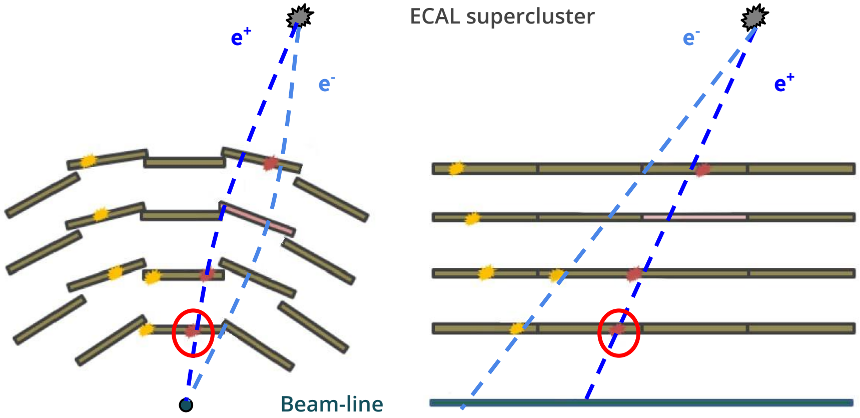
1. A path is propagated from the ECAL towards the beamline through the pixel detector assuming either e+ or e- hypothesis

Changes towards a GPU enabled seeding algorithm

Data Structures

Algorithm logic

GPU compatible utility functions



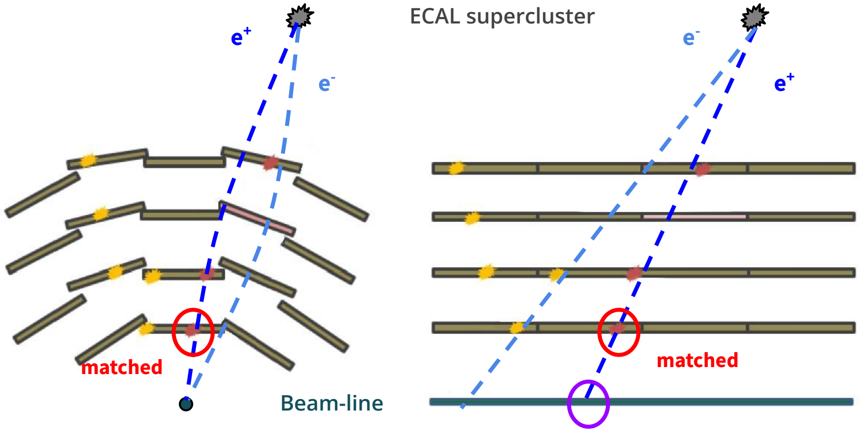
1. A path is propagated from the ECAL towards the beamline through the pixel detector assuming either e+ or e- hypothesis
2. The first hit of the seed in checked using geometrical quality criteria against the propagated track

Changes towards a GPU enabled seeding algorithm

Data Structures

Algorithm logic

GPU compatible utility functions



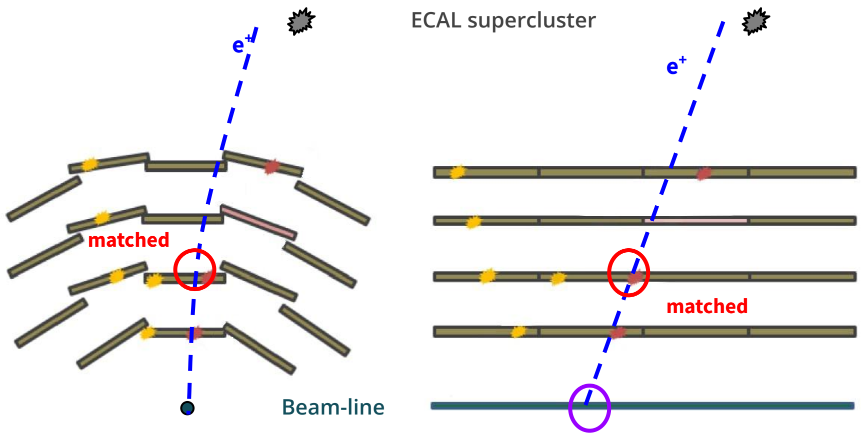
3. If a match is performed, a more precise Z position on the beamspot is determined

Changes towards a GPU enabled seeding algorithm

Data Structures

Algorithm logic

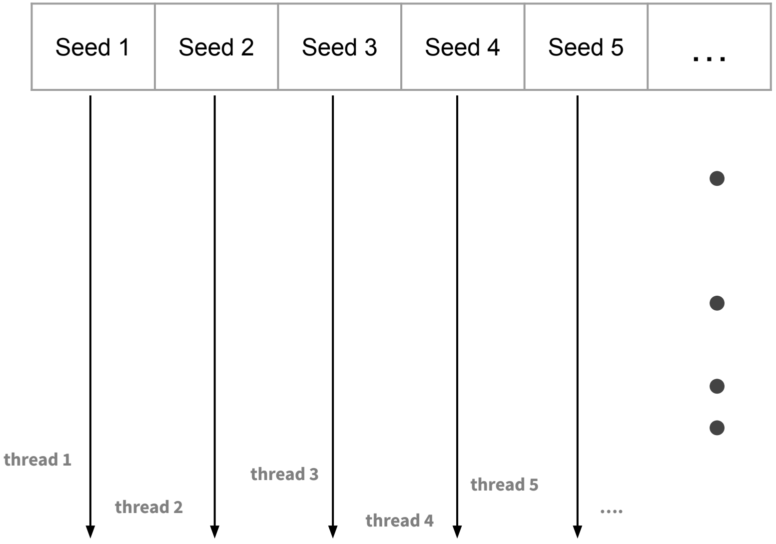
GPU compatible utility functions



3. If a match is performed, a more precise Z position on the beamspot is determined
4. Propagation is now done along the electron momentum looking for matches with additional hits
5. A seed is kept if all hits are matched

Changes towards a GPU enabled seeding algorithm

To parallelize the algorithm we assign each software thread to an initial electron seed



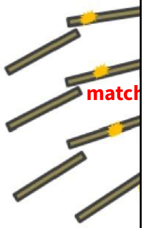
In each thread

- For each seed a loop over the ECAL superclusters and 2 charge scenarios is performed
- State of the track is updated propagating through the detector
- Quality cuts are checked
- If all hits of the seed pass the quality cuts the seed is accepted along with the matched supercluster id.

Data

ible
ons

more
nspot is
ng the
g for
hed



Utility functions

Data Structures

Algorithm logic

GPU compatible utility functions

- In order to port the algorithm described to GPU several utility functions were adapted to work both on the GPU and the CPU
  - e.g. propagation functions for propagating helix paths through the detector
  - simple structures to hold information about surfaces and trajectories
  - a simplified description of the magnetic field
- The alpaka performance portability library was utilized for these developments

Utility functions: Simplified magnetic field description

Data Structures

Algorithm logic

GPU compatible utility functions

- Previously : Full magnetic field needed to propagate the helix through the ECAL to the tracker
- Difficult to implement this on the GPU
    - Requires loading full detector geometry and material effects

Utility functions: Simplified magnetic field description

Data Structures

Algorithm logic

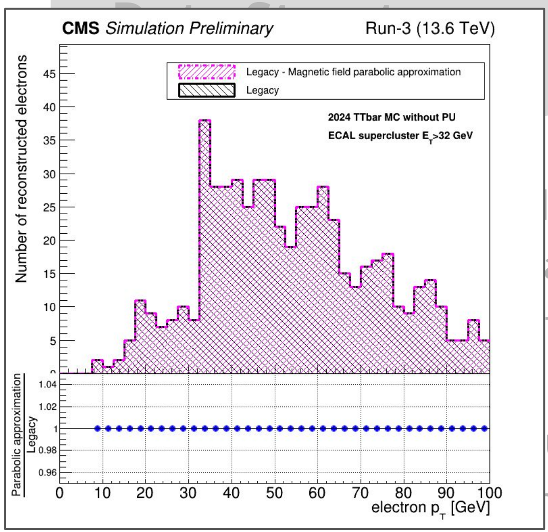
GPU compatible utility functions

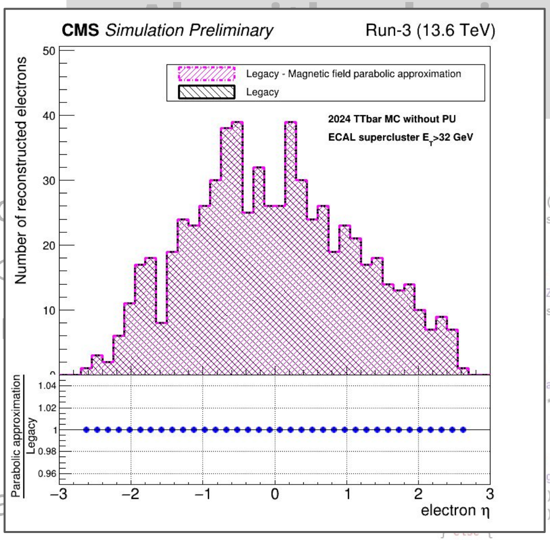
- Opted for simplified description using a parametrized parabolic approximation
  - Simple functional form that is easy to use on the GPU
  - Extended validity up to the ECAL for the electron seeding algorithm

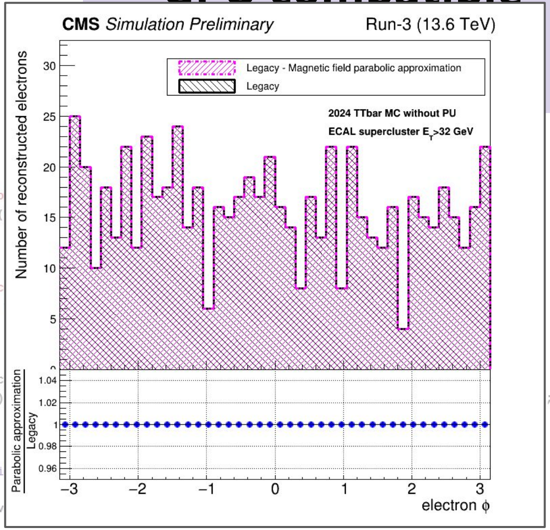
```
template <typename Vec3>
constexpr float Kr(Vec3 const& vec) {
  return Parameters::a * (vec(0) * vec(0) + vec(1) * vec(1)) + 1.f;
}

template <typename Vec3>
constexpr float B0Z(Vec3 const& vec) {
  return Parameters::b0 * vec(2) * vec(2) + Parameters::b1 * vec(2) + Parameters::c1;
}

template <typename Vec3>
constexpr bool isValid(Vec3 const& vec) {
  return ((vec(0) * vec(0) + vec(1) * vec(1)) < Parameters::max_radius2 && fabs(vec(2)) < Parameters::max_z);
}

template <typename Vec3>
constexpr float magneticFieldAtPoint(Vec3 const& vec) {
  if (isValid(vec)) {
    return B0Z(vec) * Kr(vec);
  } else {
    return 0;
  }
}
```

Utility functions: Simplified magnetic field description

GPU compatible





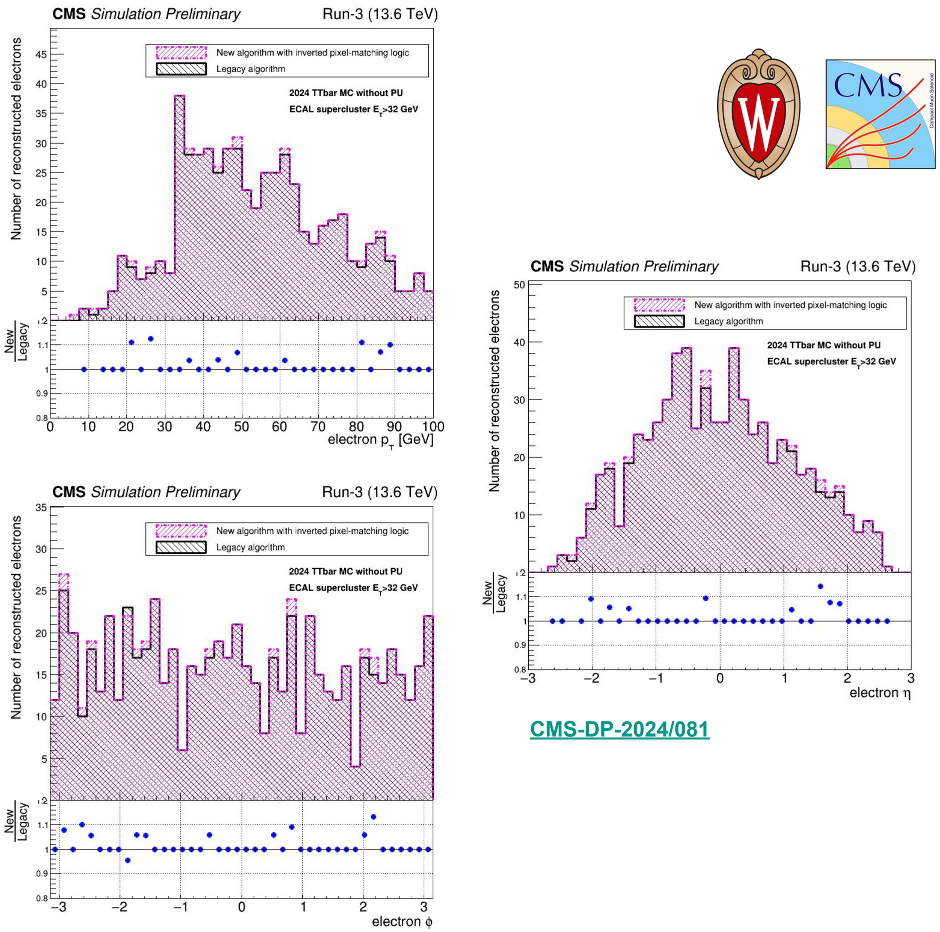


CMS-DP-2024/081

Excellent agreement between the old (legacy) approach and the new simplified approach for the magnetic field parametrization

Putting things together

- We check the initial performance of our algorithm where :
  - We use the parallelized version of the algorithm
  - We make use of the memory efficient SoAs
  - We utilize the alpaka based utility functions
  - We utilize the approximate magnetic field



CMS-DP-2024/081

Putting things together

- We check the initial performance of our algorithm where :

on of

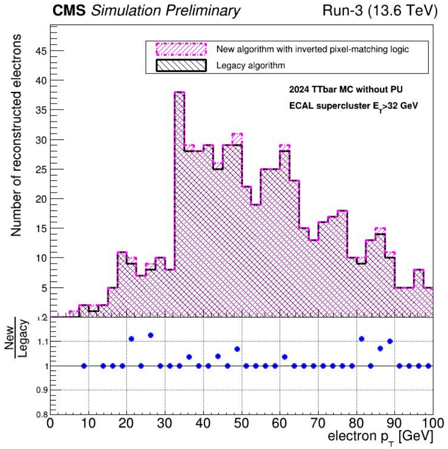
mory

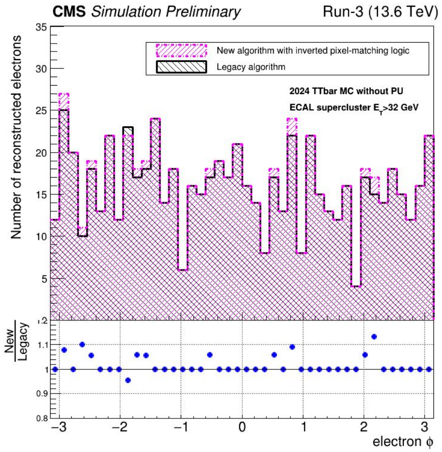
utility

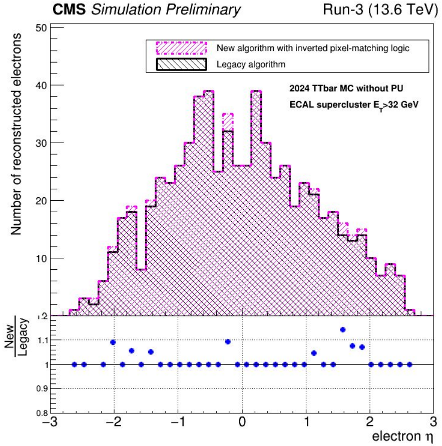
mate

magnetic field

- Very good agreement between preliminary algorithm implementation and legacy implementation
- Differences mostly due to additional quality cuts that are not currently applied







CMS-DP-2024/081

Summary

- High Luminosity LHC will pose a significant computing challenge for CMS
  - Having flexible software & taking advantage of heterogeneous resources is essential
  - Many parts of the CMS reconstruction are being redesigned with GPUs in mind with several algorithms already used for data-taking at HLT since the start of Run 3

- Electron reconstruction takes up a significant part of the overall CPU time with electron seeding being the most time consuming module

- First steps towards a GPU enabled electron seeding algorithm have been summarized:
  - Using alpaka to ensure portability and flexibility of software
  - Using CMS generic SoAs to ensure efficient memory access patterns
  - Redesigned algorithm logic to exploit the paralelizm of the GPU

- Initial results are promising with work ongoing to optimize performance !

Summary

- High Luminosity LHC will pose a significant computing challenge for CMS
  - Having flexible software & taking advantage of heterogeneous resources is essential
  - Many parts of the CMS reconstruction are being redesigned with GPUs in mind with several algorithms already used for data-taking in HLT since the start of Run 3

THANK YOU!

- Electron reconstruction takes up a significant part of the overall CPU time with electron seeding being the most time consuming module
- First steps towards a GPU enabled electron seeding algorithm have been summarized:
  - Using alpaka to ensure portability and flexibility of software
  - Using CMS generic SoAs to ensure efficient memory access patterns
  - Redesigned algorithm logic to exploit the paralelizm of the GPU

Questions?

- Initial results are promising with work ongoing to optimize performance !