Recent progress in accelerating avalanche simulation using Garfield++

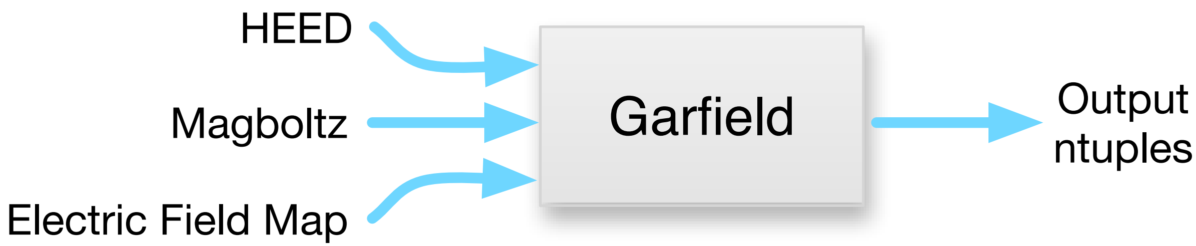
Othmane Bouhali, Ali Sheharyar

Texas A&M University in Qatar

Outline

- Simulation setup
- Processing time
- Parallelization
- Optimization
- Future work

Simulation workflow



Excessively CPU and wall time consuming (when simulating large volumes/high gain)

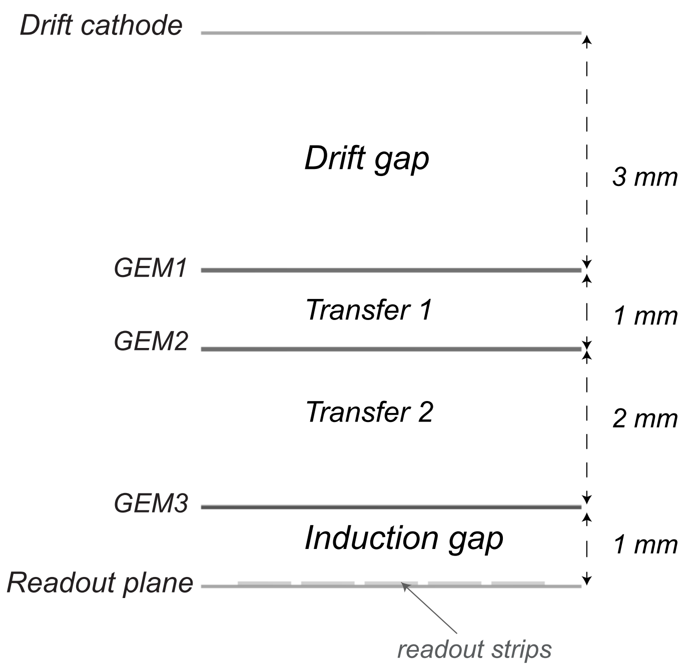
Sequential nature of Garfield

Complex electric field map

This work tackles both problems

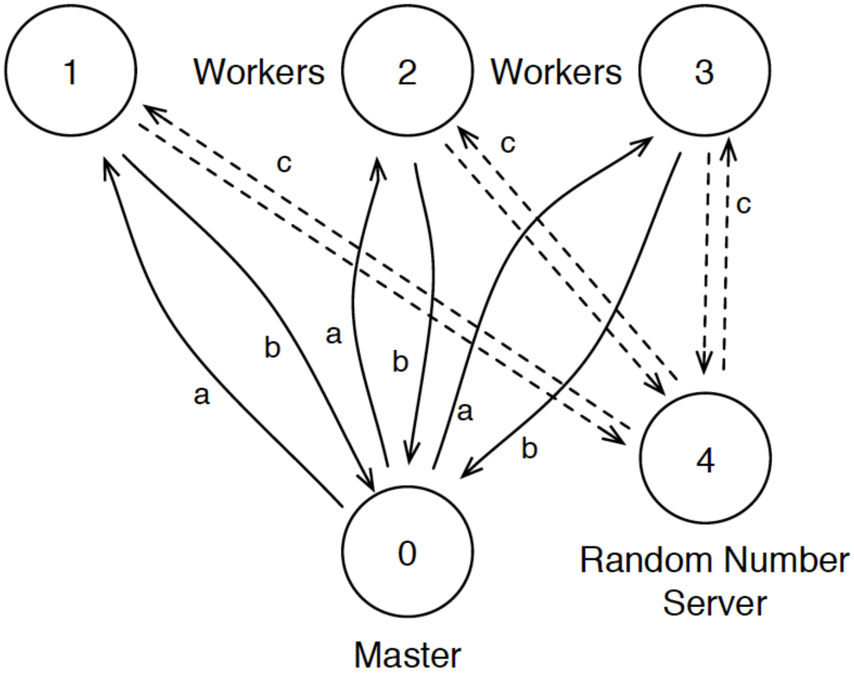
Simulation setup

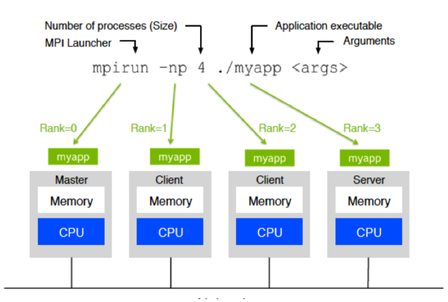
- Single GEM gap dimensions: Drift gap (1mm), Induction field (1mm)
- Triple GEM gap: Drift (3mm), TF1(2mm), TF2(1mm), Induction ()
  (similar to CMS GE1/1 configuration)
- Electric field map: ANSYS



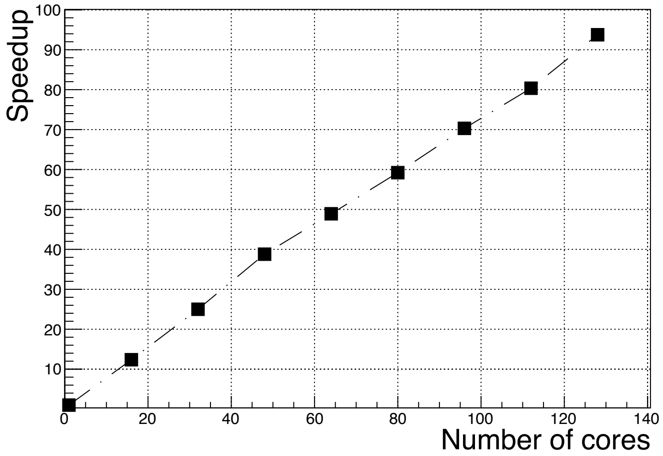
Sequential problem: event based parallelization

- Solution based on MPI
- Random number server
- Master distributes the workload and gathers results back
- Work distributed over N cores
- Using Raad and Raad2 supercomputers at Texas A&M university at Qatar



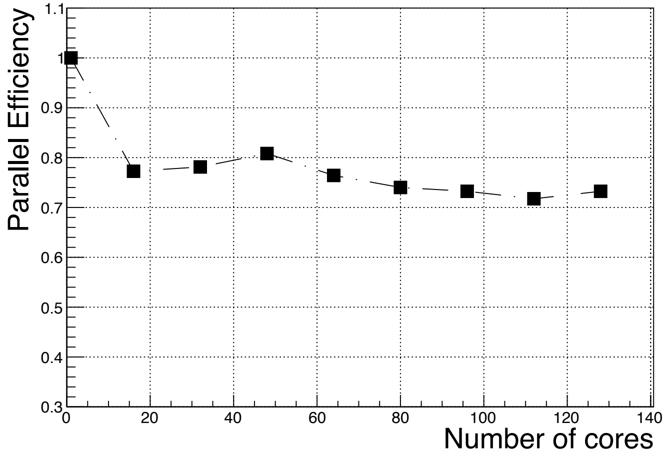


Sequential problem: event based parallelization



$$\textbf{speedup} = \frac{execution\ time\ in\ sequential\ mode}{execution\ time\ on\ parallel\ mode}$$

- Almost linear increase with the number of cores



Parallel efficiency is:

$$\frac{speedup\ factor}{number\ of\ cores\ in\ the\ parallel\ process}$$

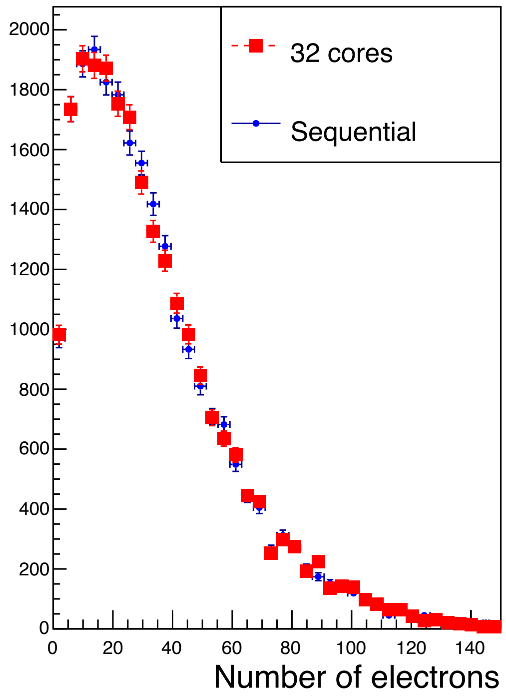
- Measures how well processors are utilized

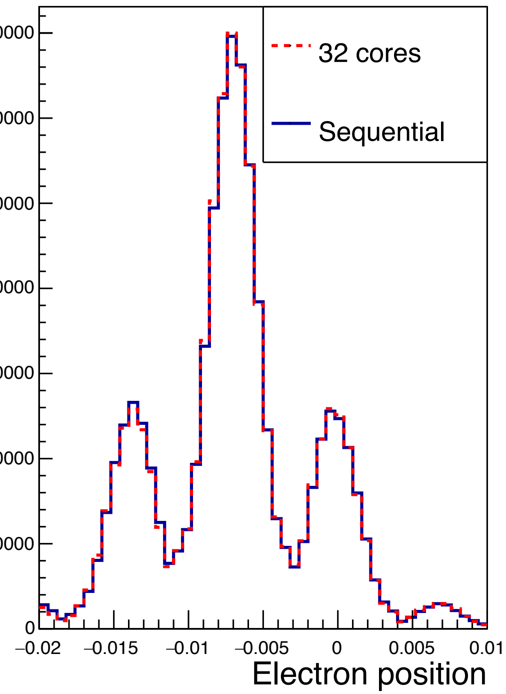
TEXAS A&M UNIVERSITY at QATAR

TEES TASC
TEES ADVANCED SCIENTIFIC COMPUTING CENTER
TEXAS A&M ENGINEERING EXPERIMENT STATION

Sequential problem: event based parallelization

Comparison: same simulated conditions

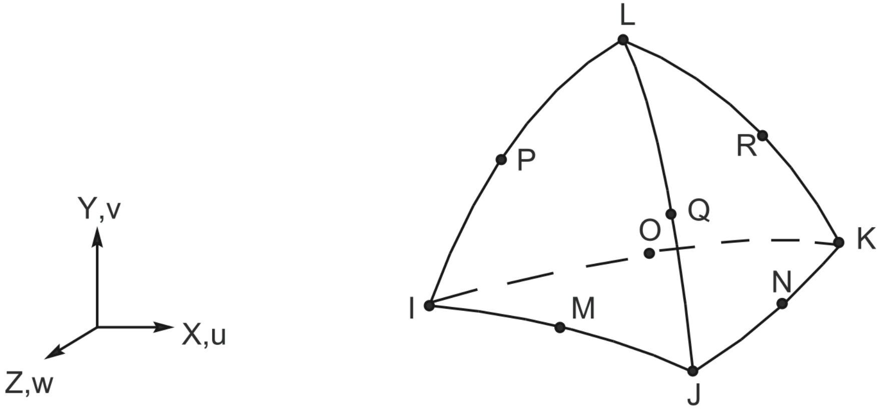


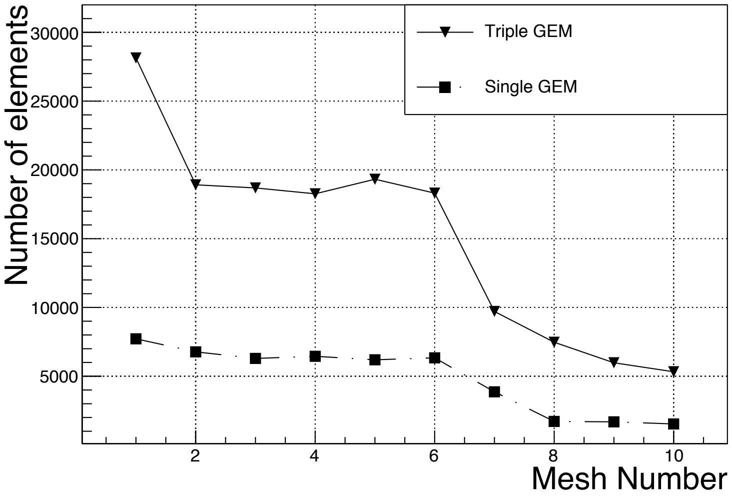


Electric field map in ANSYS

ANSYS:

- Mesh based on tetrahedral nodes
- field map mesh depends on the **accuracy** needed
- Mesh numbering: **1** (highest accuracy) and **9** (**lowest** accuracy)
- Complexity **increases** with the geometry

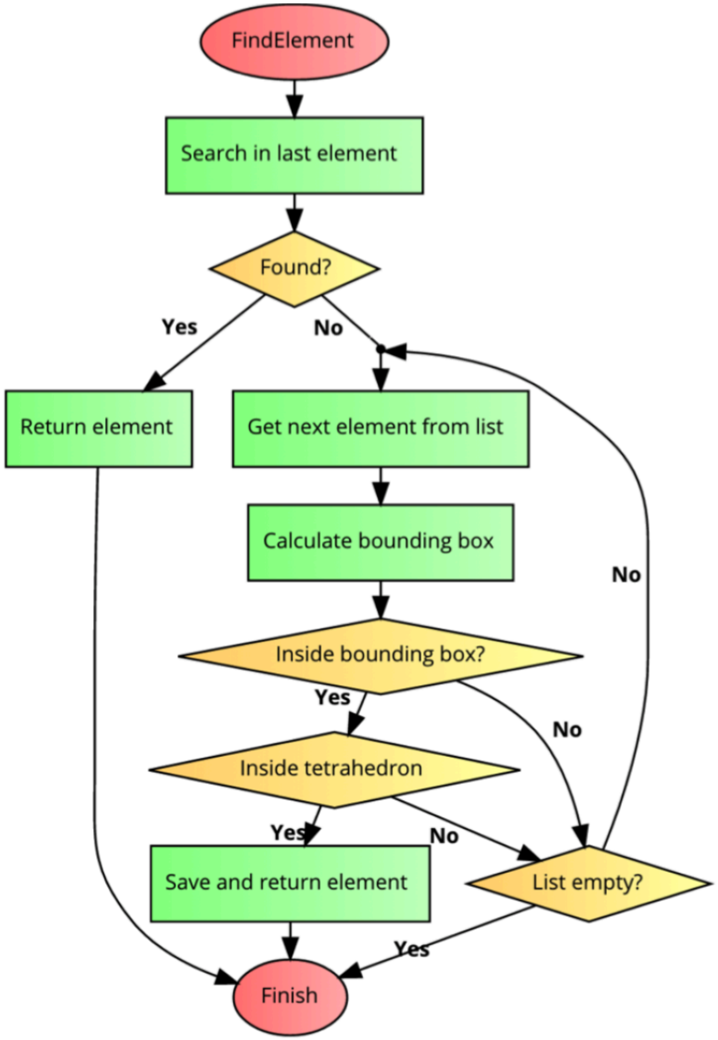




Time consuming operation

FindElement is an algorithm in Garfield++

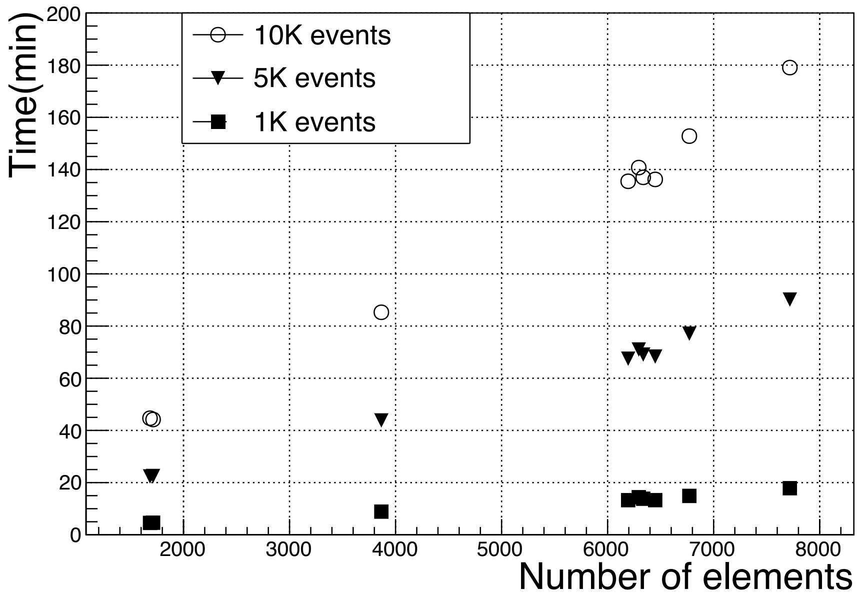
- Finds tetrahedral node containing the 3D points associated with the electron position
- **90%** of processing time is spent in this operation
- Processing time increases with mesh complexity



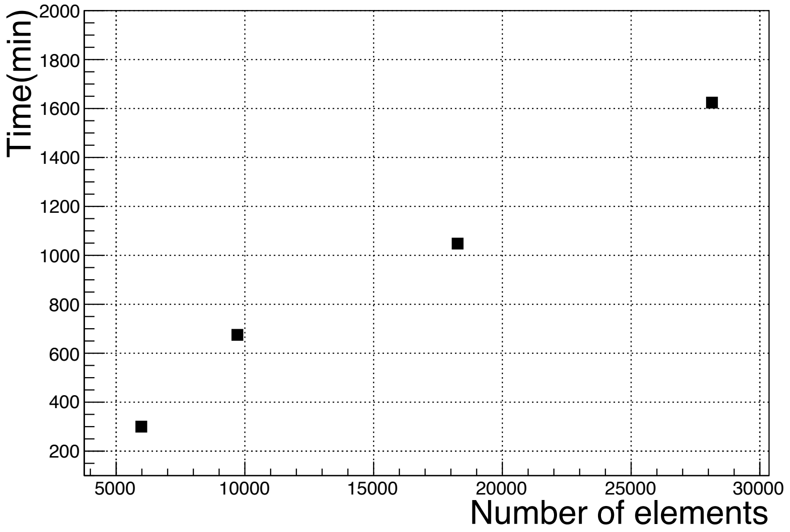
Template

- Processing time increases with mesh complexity

Single GEM



Triple GEM



→ Search technique in FindElement Needs optimization

Optimizing FinElement

Three methods are introduced:

- Caching bounding boxes
- Search over neighbors
- Spatial indexing using Octree

Caching of bounding boxes

- At each iteration the last mesh element (e position) is calculated and the program starts looking for the next location.

- A bounding box that englobes the element is defined and the program checks if the electron resides inside.
- if it is found, then it checks whether the electron is inside the mesh elements,
- if not it will go to the next element, defines the bounding box and checks again.

→ This prevents iterating over all mesh elements as it is in the original version

Search over neighbors

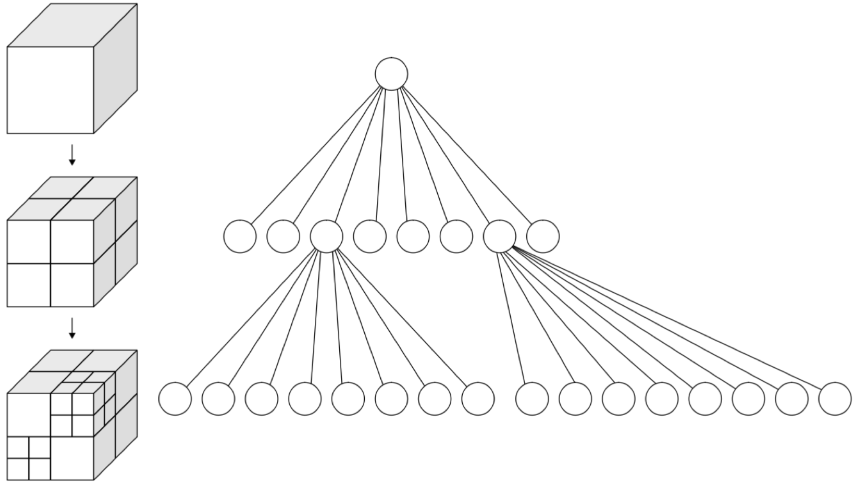
- Starting from the actual position of the electron, its next position is most likely to be within one of the neighboring elements.

- This method searches through the neighbors elements first.

- The scan over non-neighbors is performed only if the search in neighbors fails.

Search using Octree structure

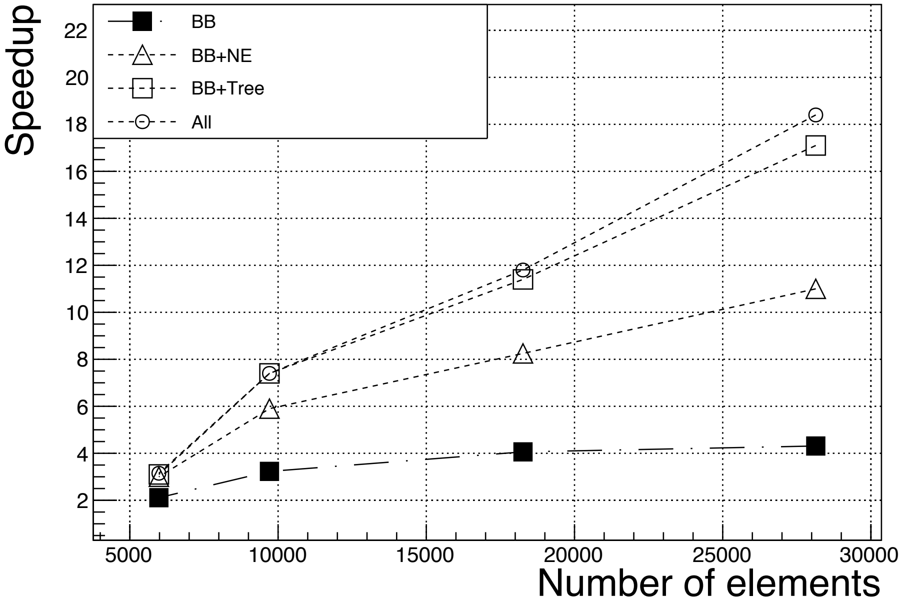
- Based on spatially indexed data structure (Octree)



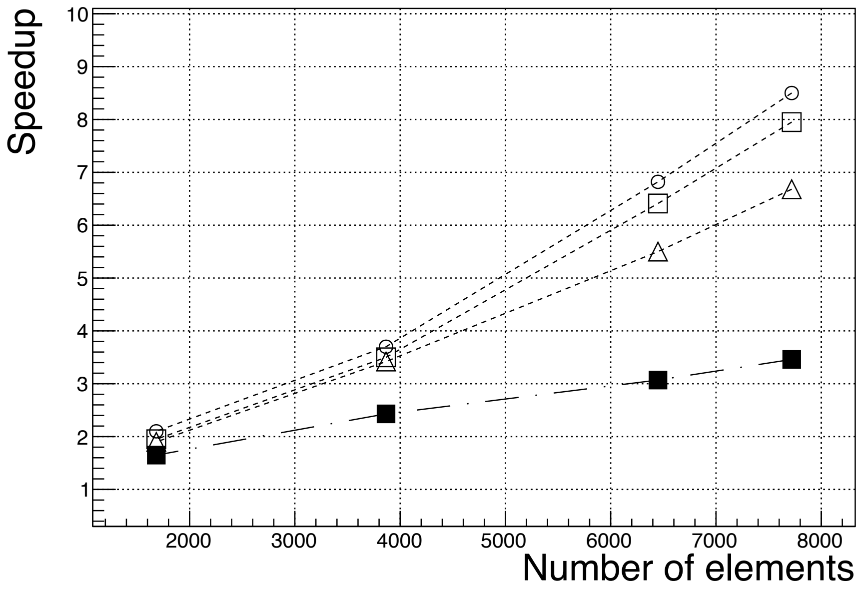
- subdivides the space in eight octants of equal dimensions and store the nodes of the tetrahedrons in a hierarchical fashion.

Optimization results

Triple GEM

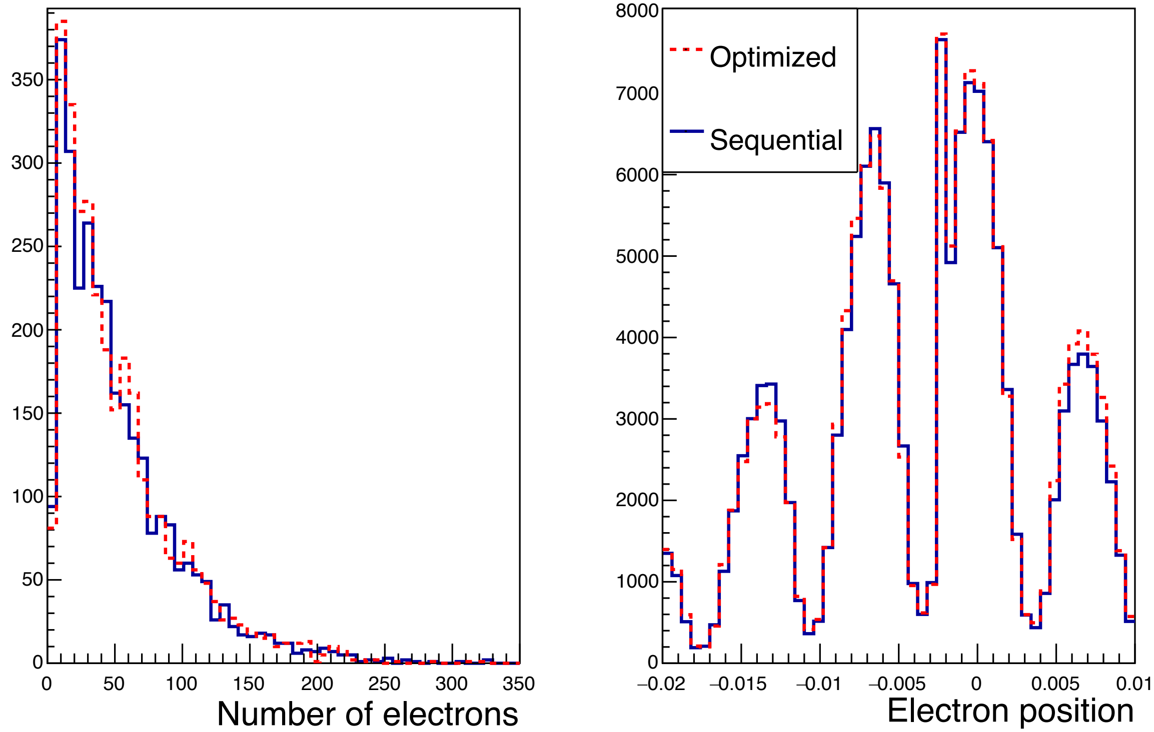


Single GEM



- Almost **20 times increase** in speed up factor
- Speedup is even **more important** in **complex geometries** (single versus triple GEM)

Comparison: optimized-sequential



- 5000 events in triple GEM with gain over 30,000 takes around 30 hours.

Summary and future work

- The parallel version provides a linear speedup factor
- The optimization of the search technique reduces the execution time by a factor up to 20
- Work published last week: **NIMA 901 (2018) 92-98. (also in proc. CHEP16)**

The github link is: https://github.com/alisheharyar/pGarfield-toolkit

Future:

- **Extend it to other field solvers**
- **Trying other hardware (accelerators): GPU, ARM…**