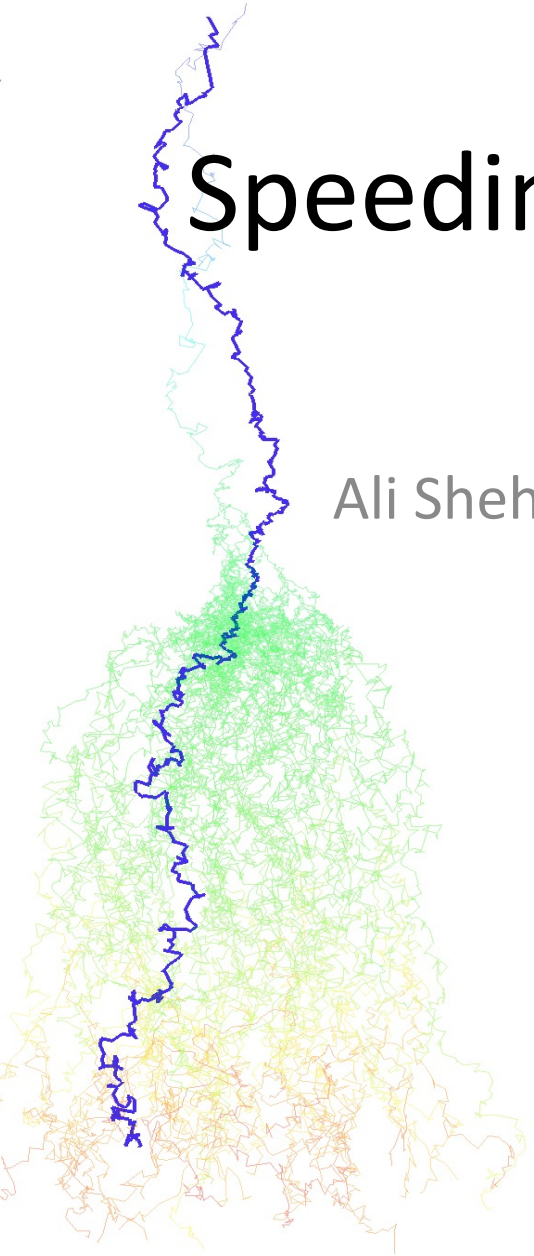


Speeding Up and Parallelizing the Garfield++

Ali Sheharyar, Othmane Bouhali, Taif Mohamed, Alfredo Castaneda

Texas A&M University at Qatar

RD51 Collaboration Meeting
October 2015

TEXAS A&M
UNIVERSITY at QATAR

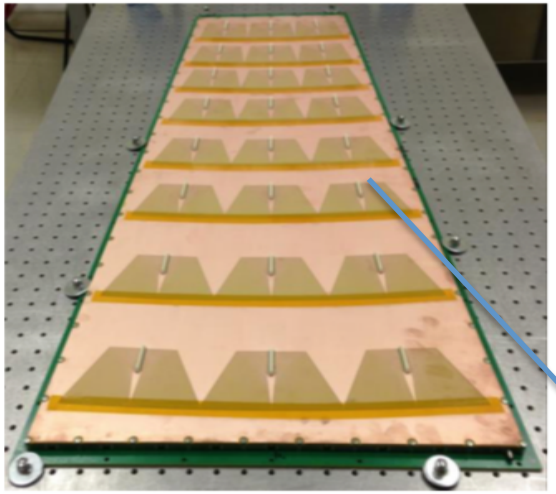
Outline

- GARFIELD++ and some applications in HEP
- Simulated processes in a gas detector
- Description of the problem
- Optimization methods proposed
- Results
- Summary on optimizations and plans
- Introduction of a parallelized version of GARFIELD++ (pGARFIELD++)

GARFIELD++

- C++ version of GARFIELD simulation toolkit http://garfieldpp.web.cern.ch
- It is a package designed for the simulation of detectors that use a gas mixture or a semi-conductor material as a sensitive medium
- It has application in several fields including High Energy Physics (HEP)
- Commonly used to estimate the performance of new detector technologies

Application in HEP



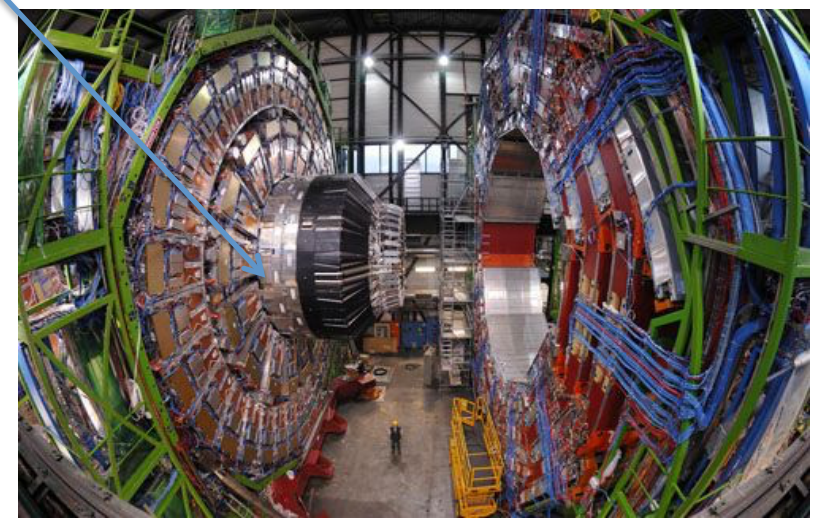
GE1/1-V-short (2014)

- GARFIELD can be used to study the performance of gas detectors, such as the Gas Electron Multiplier (GEM) detectors
- GEM detectors are well know for their excellent performance in terms of spatial and timing resolutions
- However, the use of large area GEM detectors is still a new territory, many experimental factors can influence the signal gain, therefore simulation studies are much needed

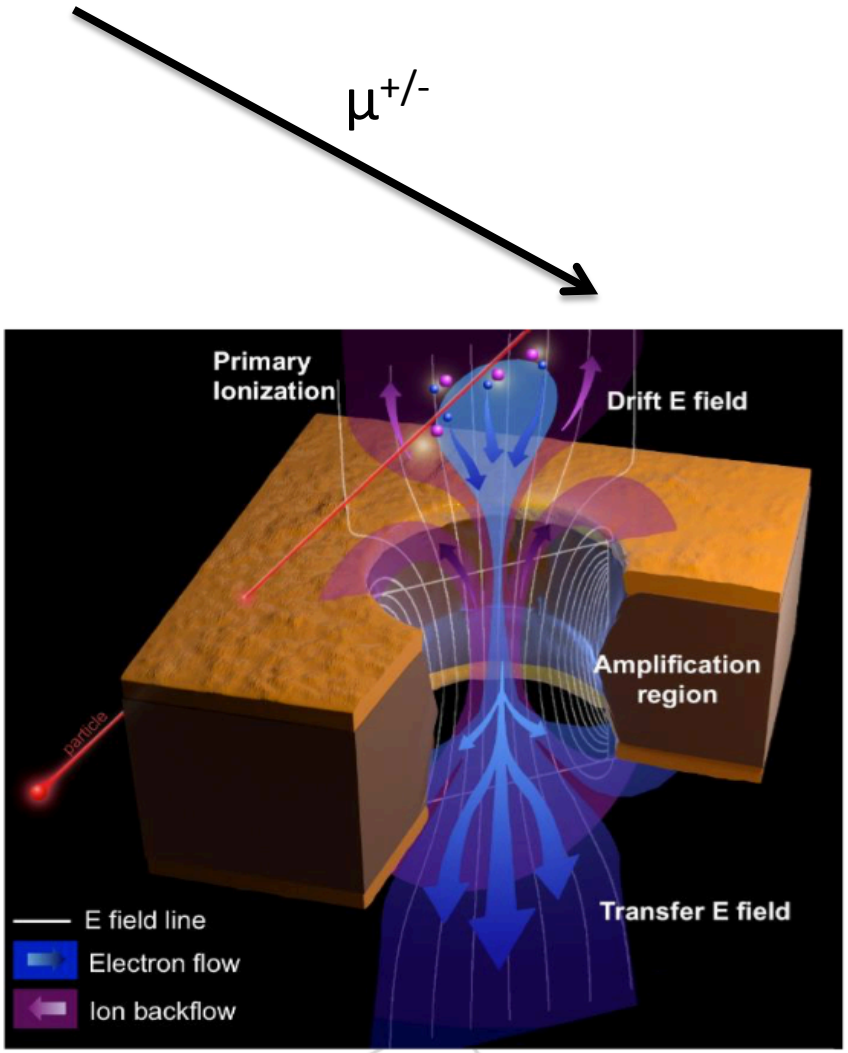
CMS experiment is planning to install GEM detectors as part of its upgrade of the muon system in order to afford the intense particle flux in the high-luminosity scenario.

More details in talk given by Brian Dorney

https://agenda.infn.it/contributionDisplay.py?sessionId=2&contribId=66&confId=8839

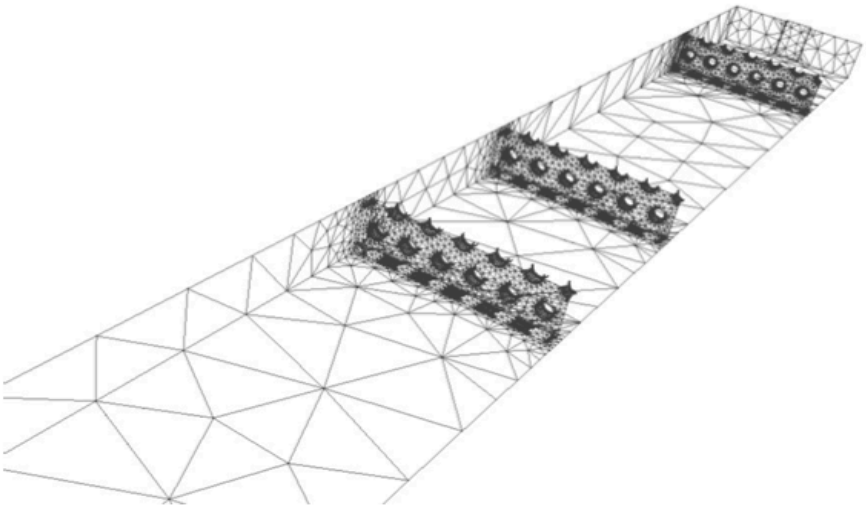


Simulated process in a GEM



- High energy particles (i.e. muons) interact with atoms and molecules in the sensitive medium (gas mixture in case of a GEM)
- During the ionization process free electrons are released and accelerated by the use of an electric field
- Those free electrons collide with further atoms releasing more electrons (amplification region)
- As a result a chain reaction (also known as electron avalanche) is created

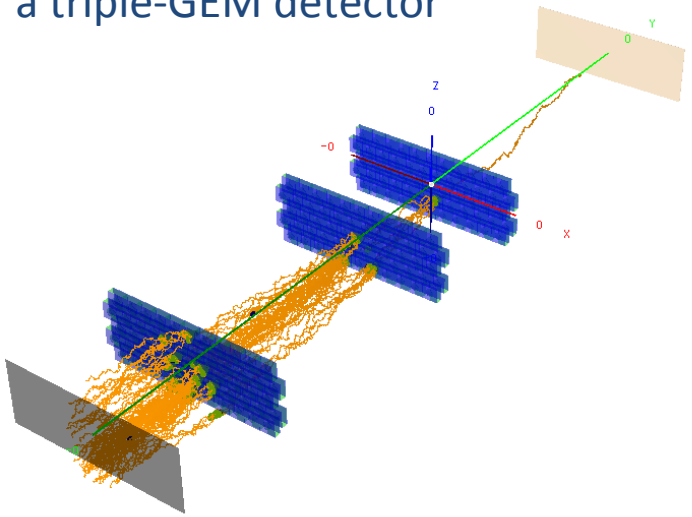
Tracking of particles in a GEM detector



- Electron position is influenced by the description of the electric field
- The electric field is calculated using external packages (i.e. ANSYS)
- A tetrahedral mesh is used to described the electric field
- The finer the mesh the longer the simulation time

- In a triple-GEM there are three amplifications stages
- Signal (from the electron avalanche) is collected at the readout board

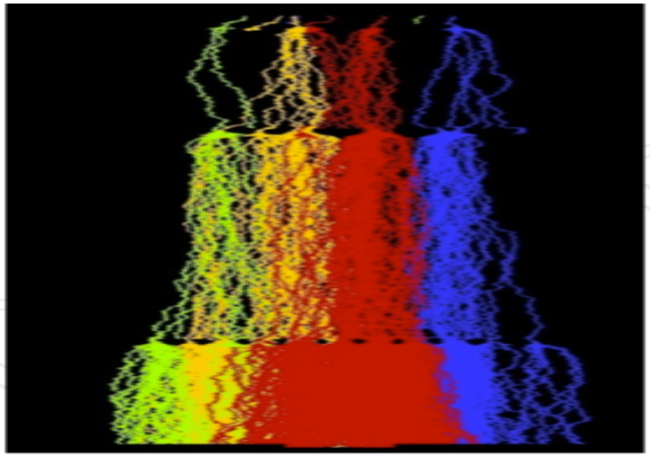
Tracking of a single electron traversing a triple-GEM detector

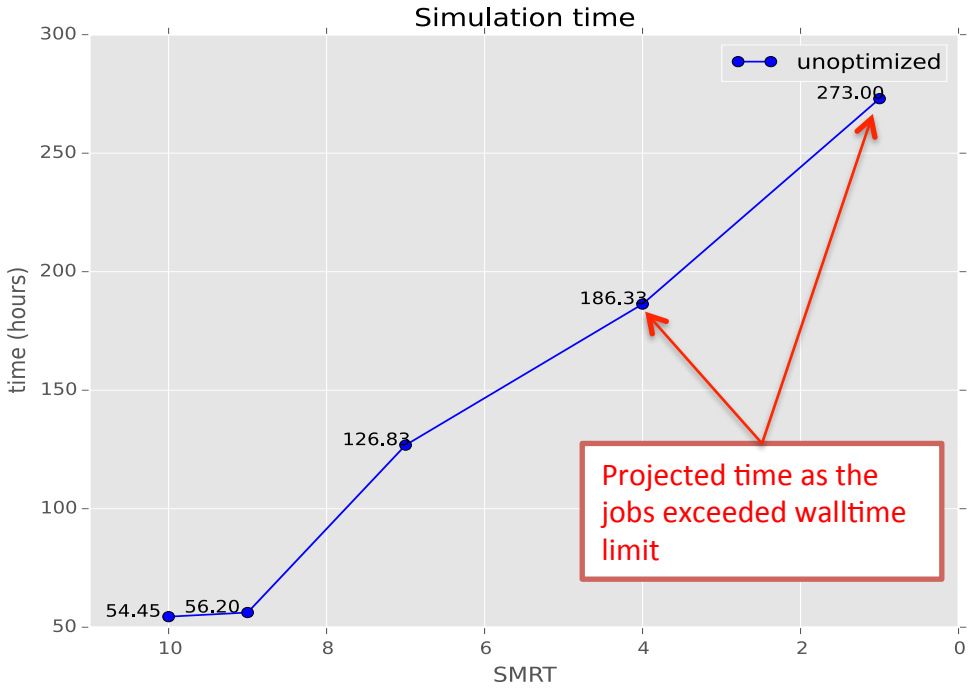


Description of the problem

- Due to the complexity of the processes the simulation can take days or even weeks
- For certain conditions (high voltage and fine mesh) the simulation takes so much time and computing resources that a timely production of results is not feasible
- The supercomputing facilities allow only limited walltime for serial jobs.

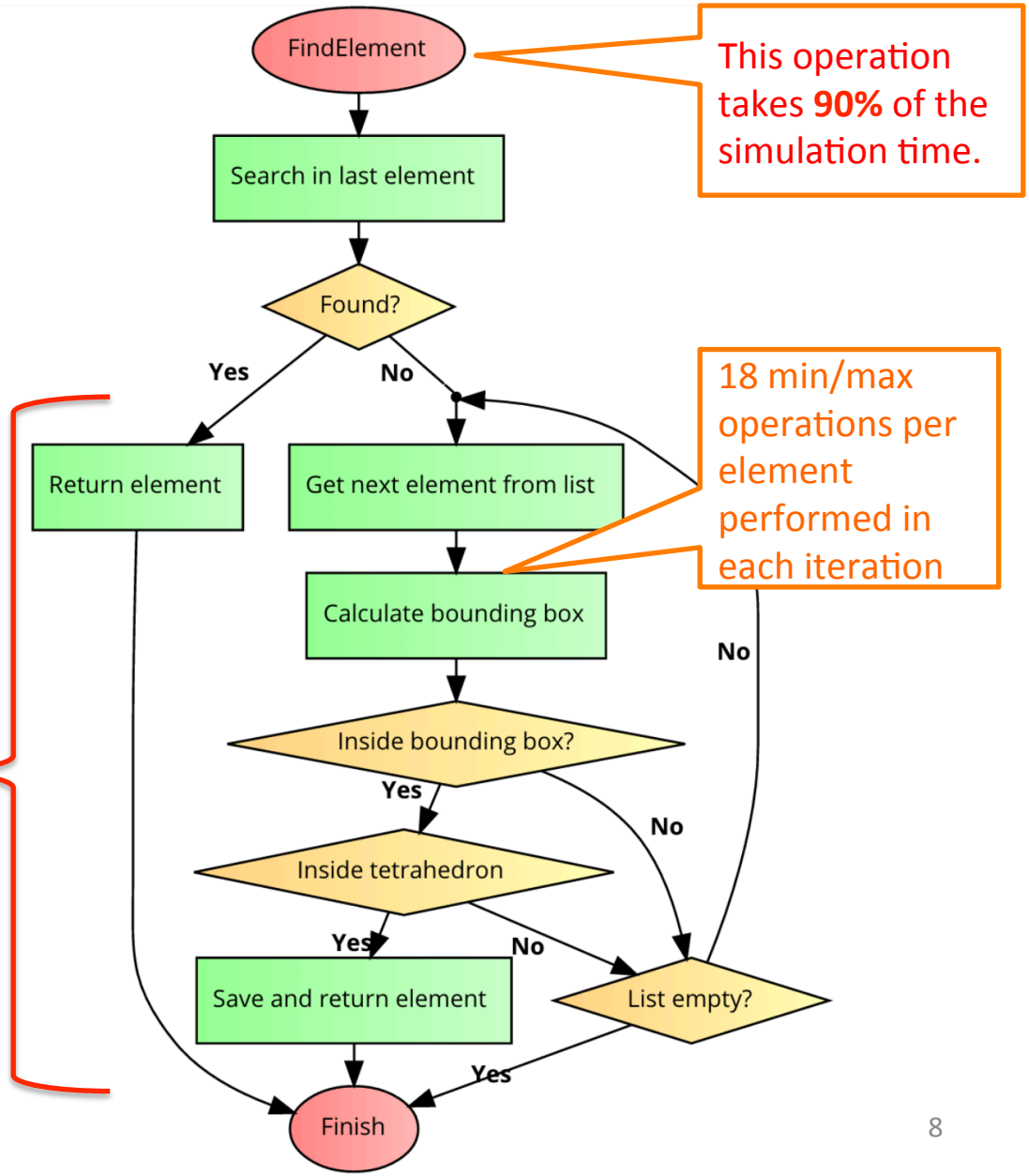
Simulation of electron avalanches using a high voltage





GARFIELD++ FindElement algorithm

- FindElement is an algorithm part of the official garfield++ package
- FindElement finds the tetrahedral in the field mesh containing the 3D point associated with the particle position

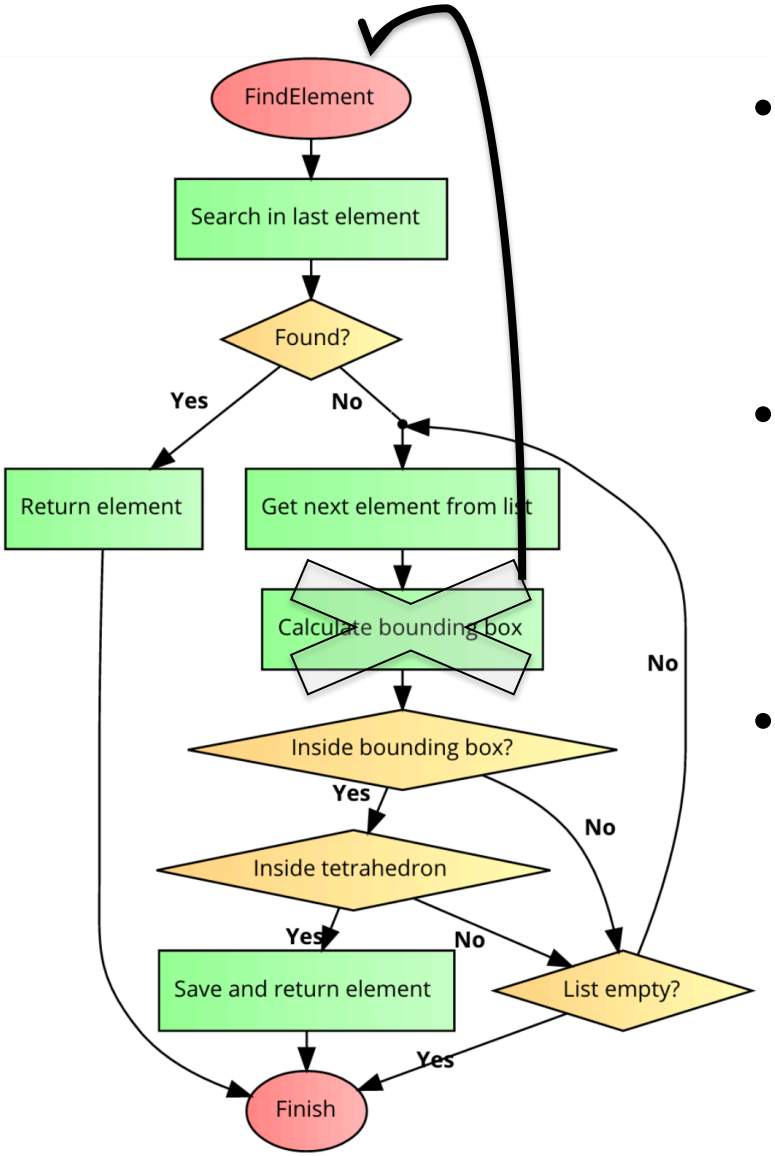


Optimization methods

Three optimization methods were studied:

1. Caching of bounding boxes.
   - Removed 18 min/max operations
2. Element search using a Tetrahedral Tree data structure.
   - Computational complexity: O(N) → O(log N)
   - Best case: O(1)
   - Worst case: O(log N)
3. Search through neighbors

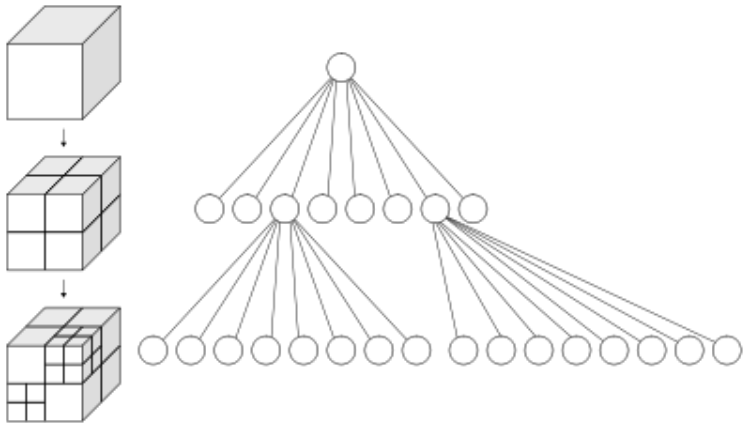
Caching of bounding boxes.

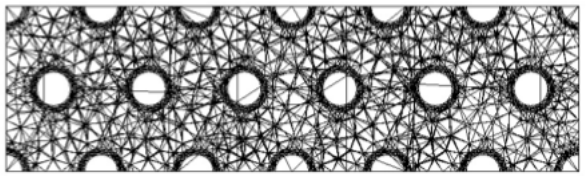


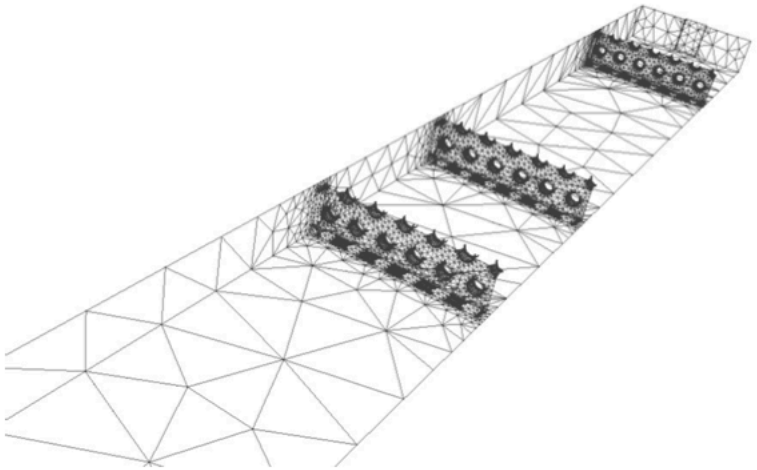
- The catching bounding box step is moved to the beginning of FindElement
- This reduces 18 min/max operations that were performed in each iteration
- This change already speeds up the simulation 5x

Tetrahedral Tree

- Based on the Octree
- Used to partition a three dimensional space by recursively sub-diving it into eight octants
- Improves the spatial indexing of nodes (vertices) and elements (tetrahedrons).
- Each leaf block stores a set of nodes (vertices).
- And also a set of tetrahedrons whose BBs intersects with the block's BB.
- More speedup expected for high resolution meshes.



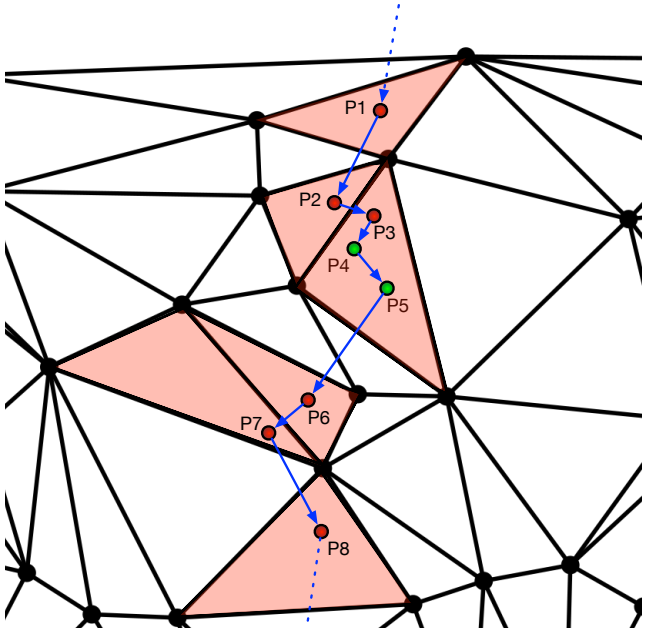




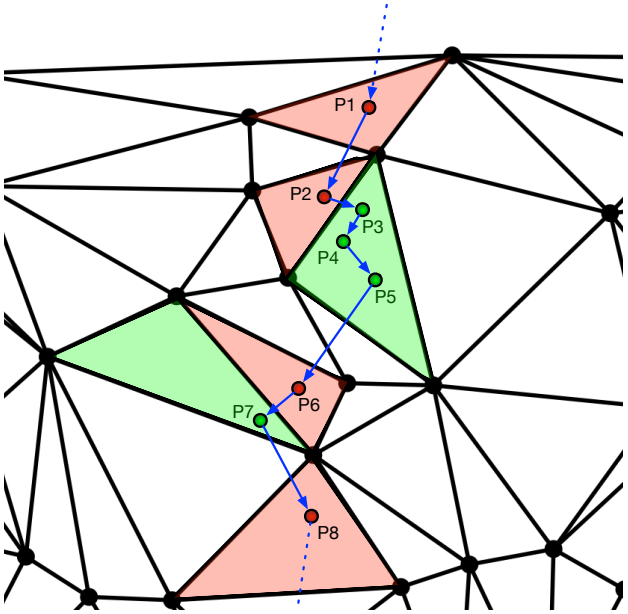
Search through Neighbors

- In the normal searching algorithm, if the last element does not correspond to the position of the particle the algorithm starts from a next element in the list
- In our optimized way we search first for neighbors where most likely the element will be found

Un-optimized



Optimized



Benchmarking

- 4 optimized cases were studied and compared to the un-optimized case:
  1. Cached bounding boxes
  2. Cached bounding boxes + neighbors
  3. Cached bounding boxes + tetrahedral tree
  4. Cached bounding boxes + neighbors + tetrahedral tree

- For 2 detector configurations.
  - Single GEM
  - Triple GEM

Benchmarking

- Single GEM
  - 10000 events
  - SMRT: 1-10
  - Jobs: 45 (9x5)
- Triple GEM
  - 1000 events
  - 2538 volts: SMRT: 1,2,3,4,5,7,9,10 (8 meshes)
  - 3243 volts: SMRT: 1,4,7,9,10 (5 meshes)
  - Jobs: 65 (8x5)
- Total jobs on RAAD: 110

Running at Texas A&M at Qatar cluster (RAAD)

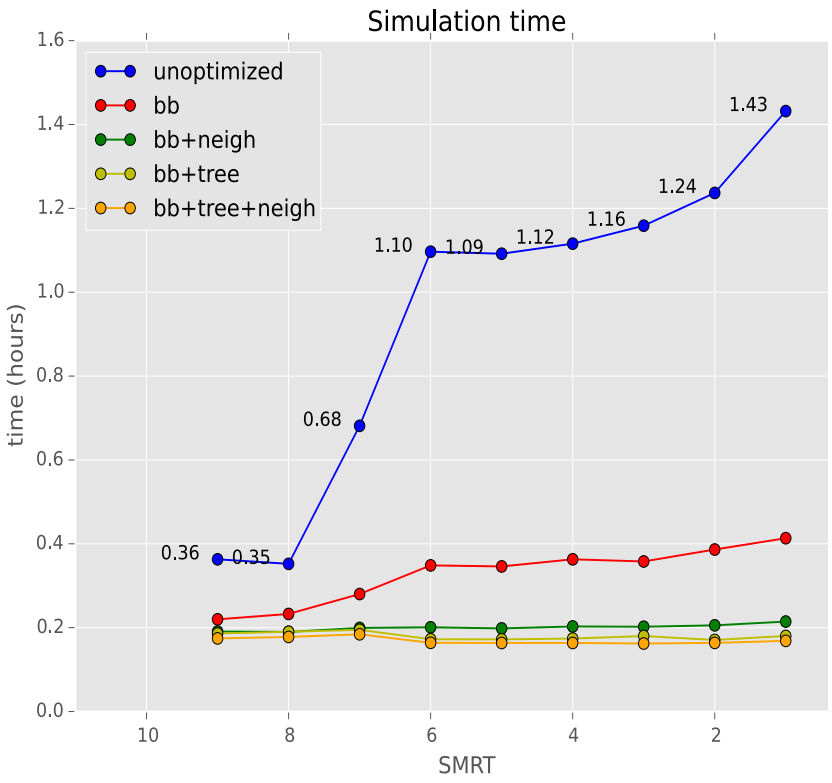
RAAD specs:

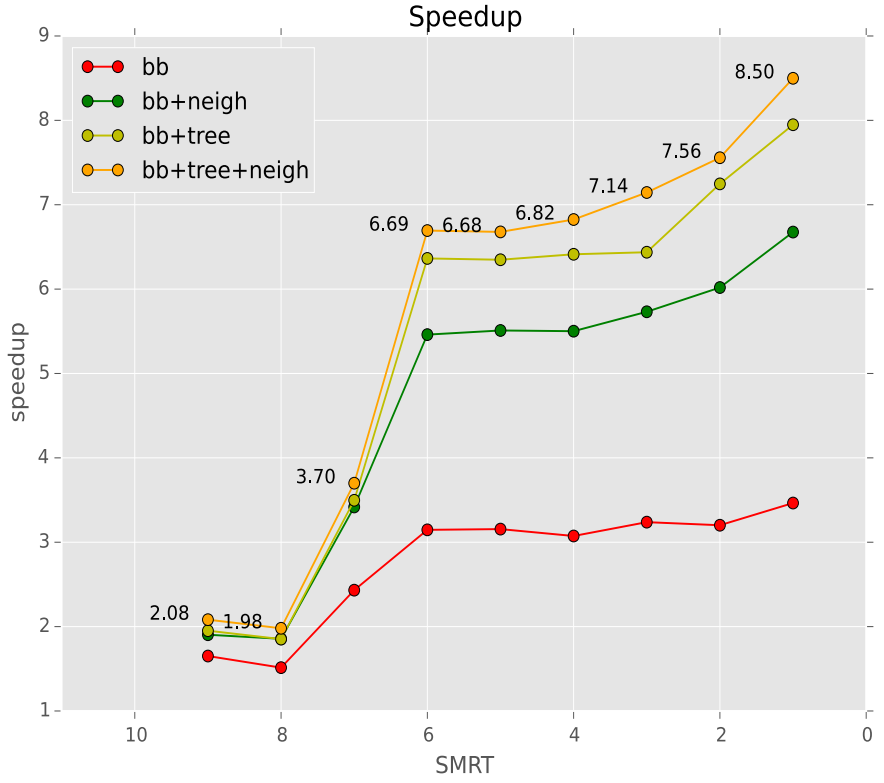
- ***2,200 CPU cores***
- ***12 Nvidia TESLA K20 (Kepler) GPUs, each with 2,496 CUDA cores***

Results

Single GEM

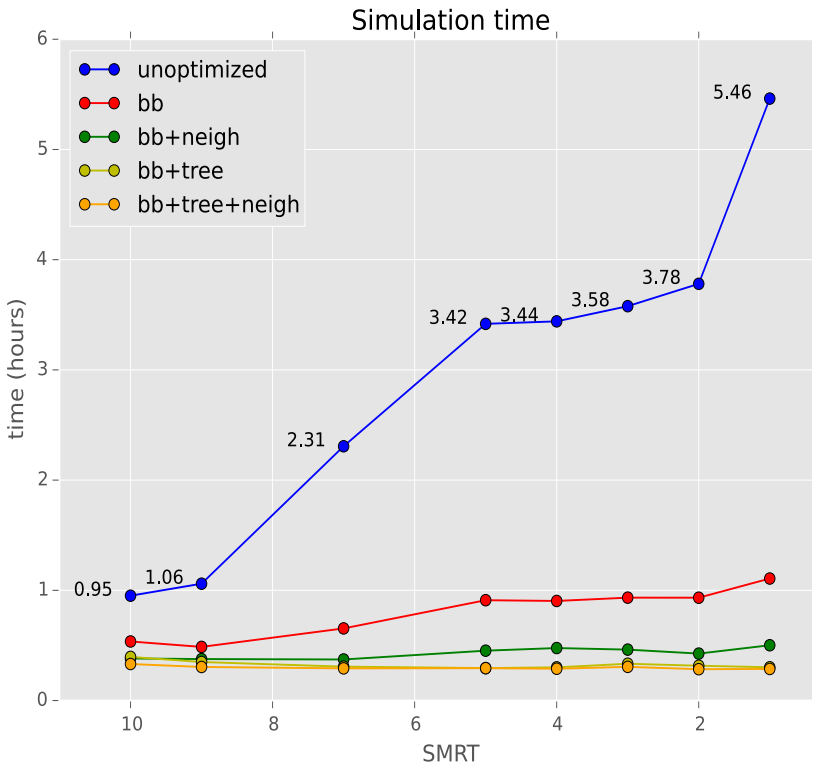
10,000 events, 9 meshes

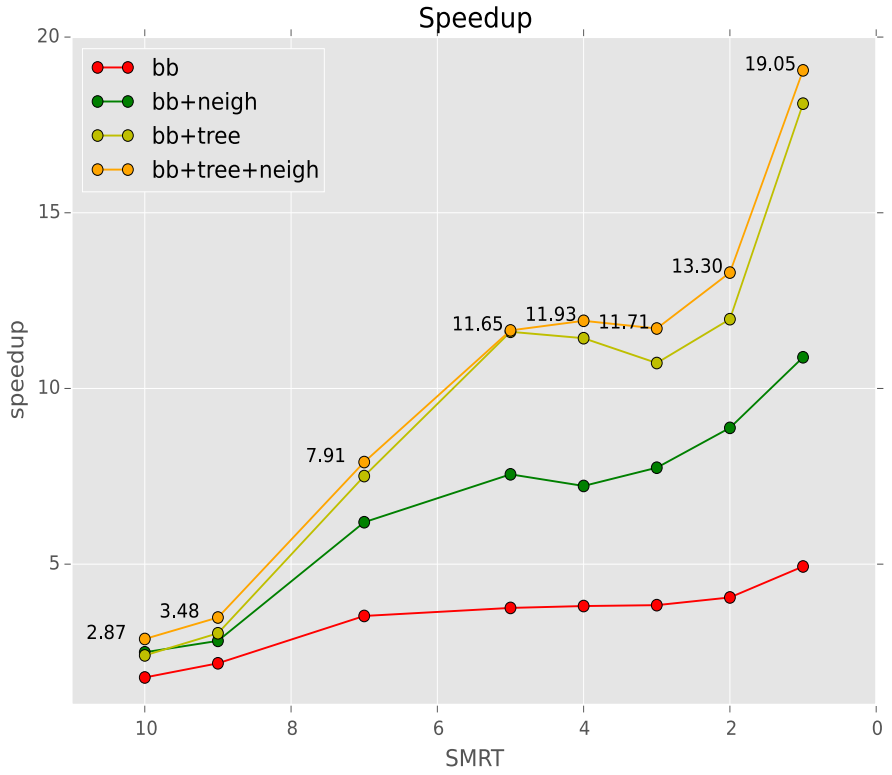




Triple GEM, 2538 volts

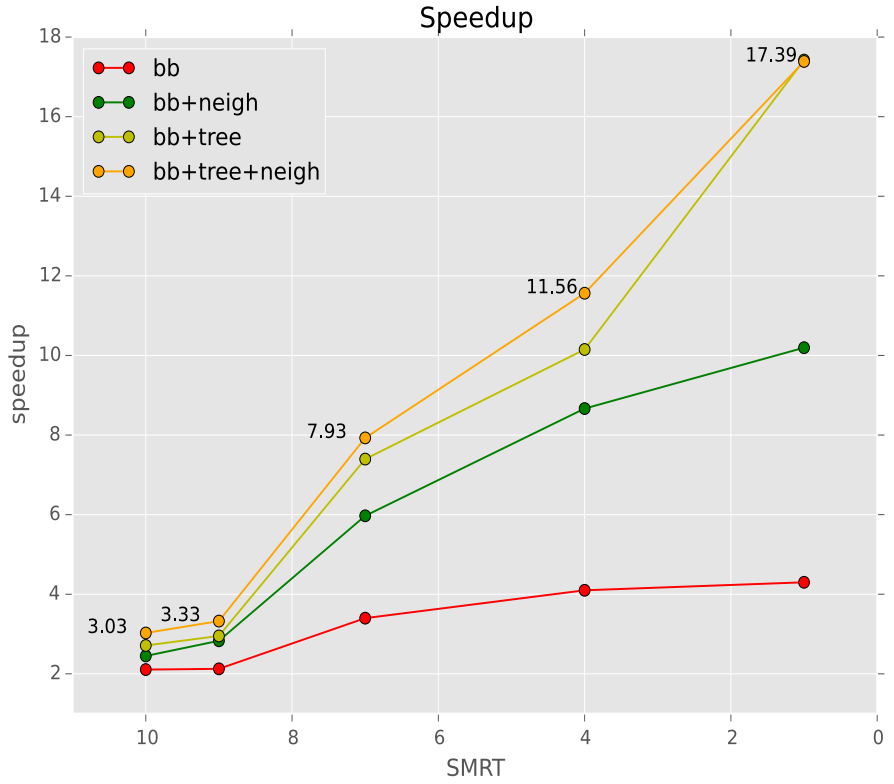
1,000 events, 8 meshes





Triple GEM, 3243 volts

1,000 events, 5 meshes



Projected based on:

1: 606 events

2: 898 events

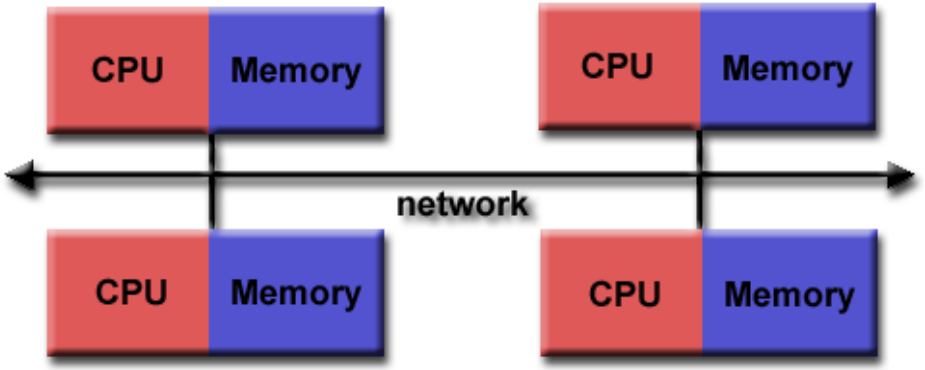
Mesh details: course to fine →

Summary on Optimization

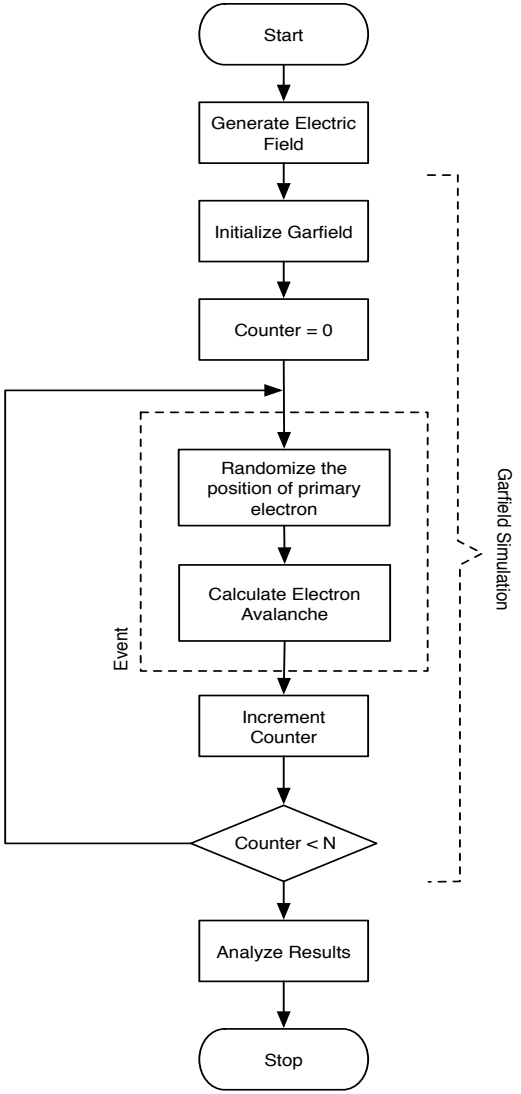
- Several improvements to the GARFIELD++ algorithms were studied
- The maximum speedup factor is 19x which is reached combining the three optimization methods
- The bounding box caching code (speed up of 5x) has already been pushed to the Garfield++ SVN repository.
- Other optimizations will be checked in soon.

Parallelization of GARFIELD++

- Need to adapt GARFIELD to a parallel programming framework.
- Shared memory or distributed memory architecture?



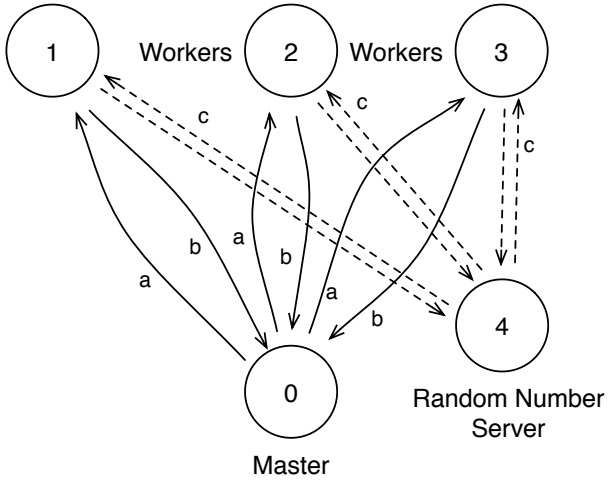
Distributed memory architecture

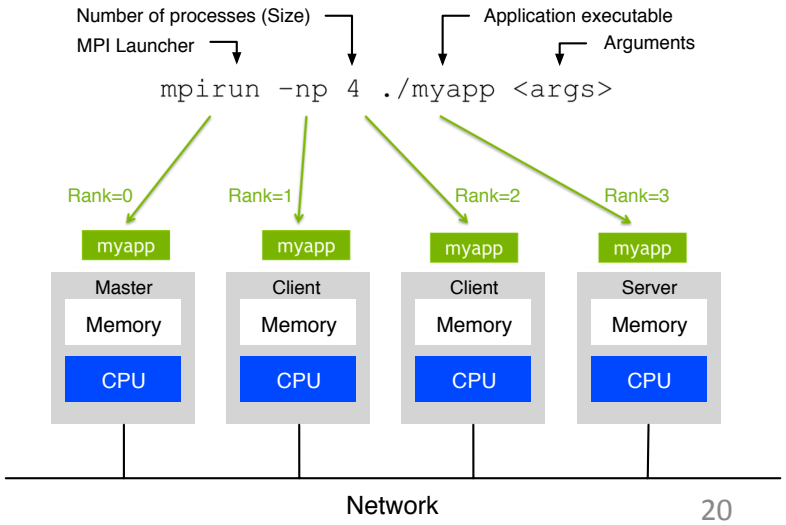


Serial simulation workflow

Parallel Garfield (pGarfield++)

- Based on Message Passing interface (MPI) architecture
- Random number generation
- Master distributes the workload and gathers the results back.

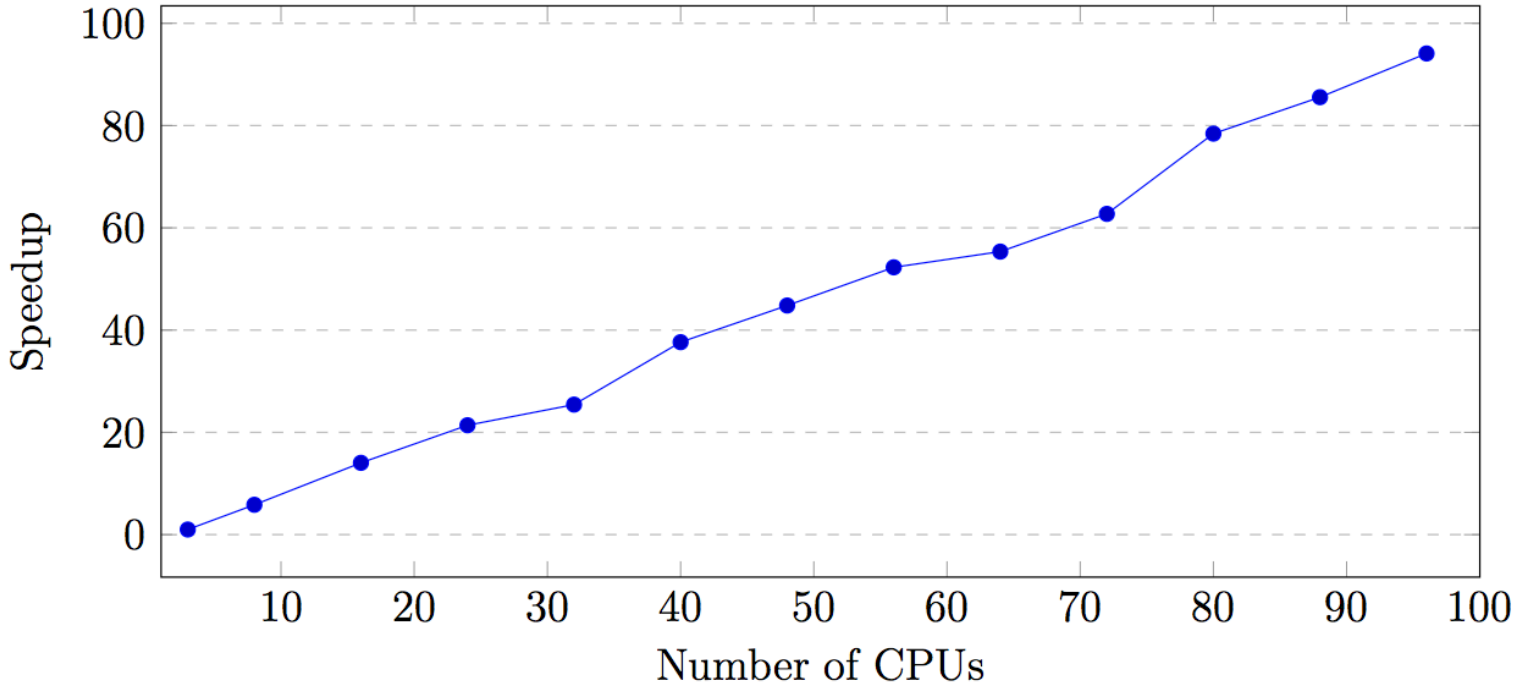




Performance Results: Speedup

Performance was evaluated on the HPC cluster at Texas A&M University at Qatar. The HPC cluster (named RAAD) is a 42+ TFLOP, 2208-core Intel Xeon system.

$$Speedup = S_p = \frac{T_s}{T_p}$$



Conclusions

- With the combination of the algorithm optimizations and the parallelization of GARFIELD++ we can reduce substantially the simulation time
- The simulation output is not affected by these optimizations, preliminary results show consistent results between the un-optimized and the optimized versions (within statistical uncertainties)
- Some of these changes already propagated into SVN
- We will continue working on our optimization studies

This work was supported by the Qatar National Research Fund under the project NPRP-5-464-1-080