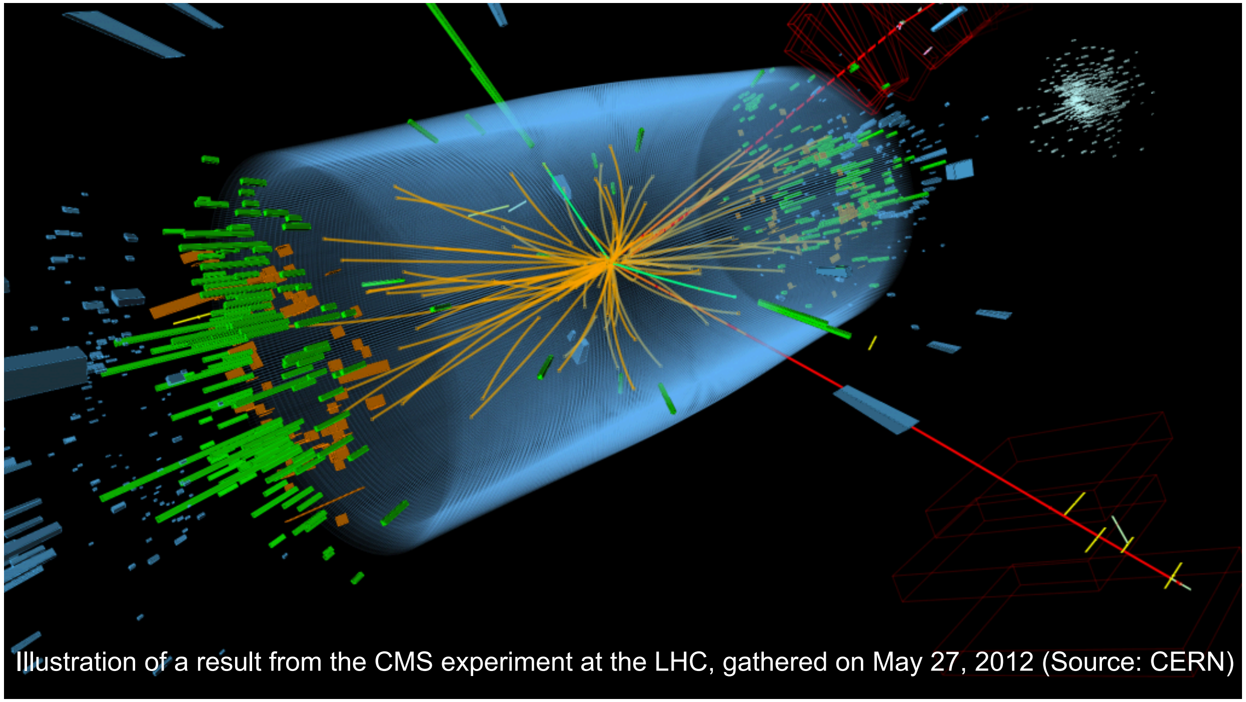
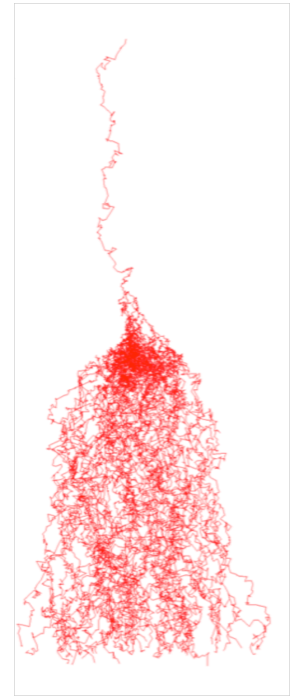
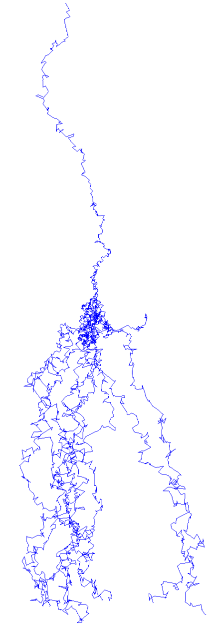


Illustration of a result from the CMS experiment at the LHC, gathered on May 27, 2012 (Source: CERN)





Speeding Up the Garfield++

11-June-2015

Ali Sheharyar (Texas A&M University at Qatar)

Dr. Othmane Bouhali (Texas A&M University at Qatar)

Problems with Garfield simulations

- Simulations take too much time.
  - … sometimes may take several days or even weeks.
- Complex and large experiments are not feasible.
  - Such as involving larger electric fields, higher voltage, gains and large number of events.

How to Speed Up?

- Three approaches:

1. Optimization of the serial Garfield++
2. Event-level parallelism
3. Track-level parallelism

- We are working on the 1st and 2nd approach currently. 3rd approach has been kept as future work. We are considering the use of GPU to simulate the individual electrons/tracks. For now, let us talk about the current work.

1st Approach

Optimizing the Garfield++

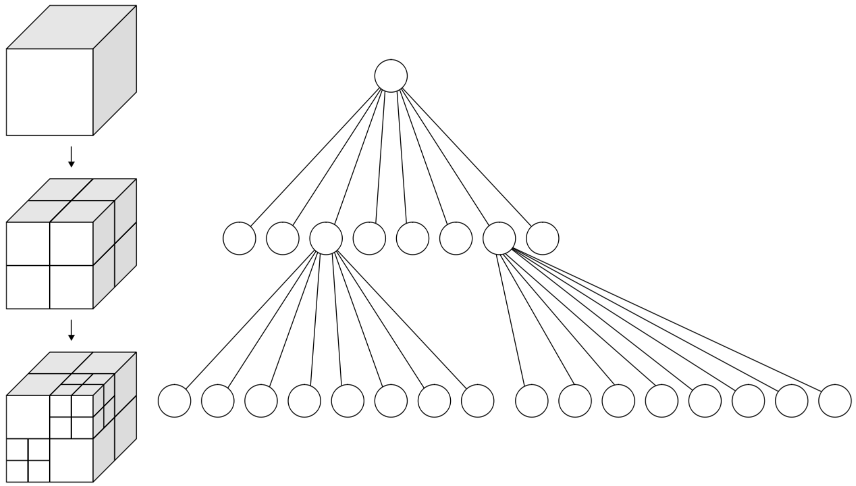
Optimize the serial implementation of the Garfield to speed up the calculations.

Observations

- The Garfield code was profiled using GNU profiler (gprof)
- It has been observed that almost 90% of the time is spent in finding the element in the electric field corresponding to a given 3D location.
- The element search is linear O(N). Garfield stores all elements in a linear array.
- If the given point is not found in the last found element, the search again restarts from the beginning of the element list and takes O(N).

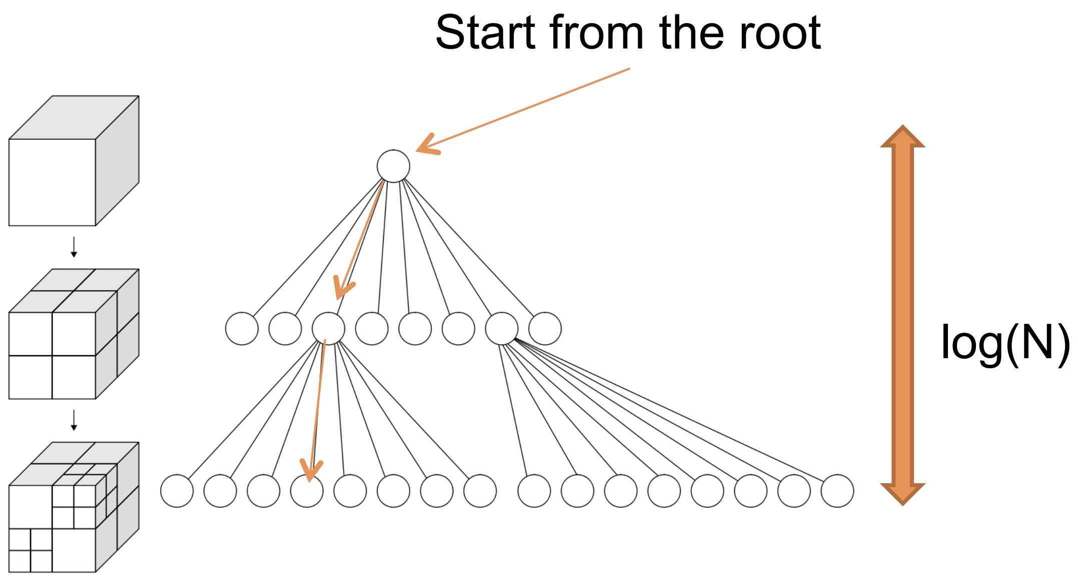
Optimization 1: Use spatially indexed data structure

- Replace the linear array data structure with a spatially indexed data structure such as PR (Point-Region) Octree.



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

Searching through the Octree



- The search time is reduced from O(N) to O(logN)

Image source: Wikipedia

Optimization 2: Search through neighbors

- Currently, if a given point is not found in the last found element, the search begins from the beginning hence taking O(N) [The octree reduces it to O(logN)].
- Considering the behavior of electron tracks, the next query position is expected to be around the previous position.
- If the new position is not found in the last found element, then it would be most likely in one of its neighbors.
- Each tetrahedron has four (4) neighboring tetrahedrons who share the same face.
- Searching the neighboring elements is a constant time operation.

Initial results

- Number of events: 100
- Scenario 1:
  - Number of elements: 8K
  - Number of nodes: 14K
  - Original time: 4m 12s
  - New time: 41s
  - **Speedup: 6.14**
- Scenario 2:
  - Number of elements: 135K
  - Number of nodes: 217K
  - Original time: 6m 22s
  - New time: 14s
  - **Speedup: 26.36**
- The speedup is expected to be greater for larger electric field meshes. The benchmarking is the work in progress

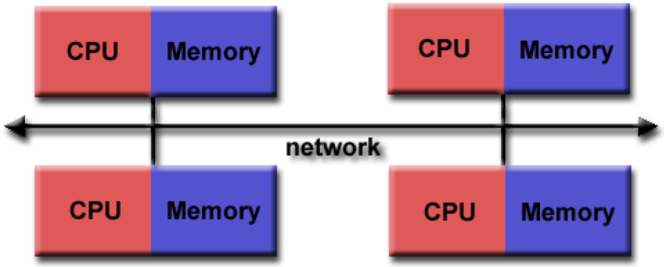
2nd Approach

Event-level parallelism

Distribute the simulation of multiple events over multiple processes using MPI

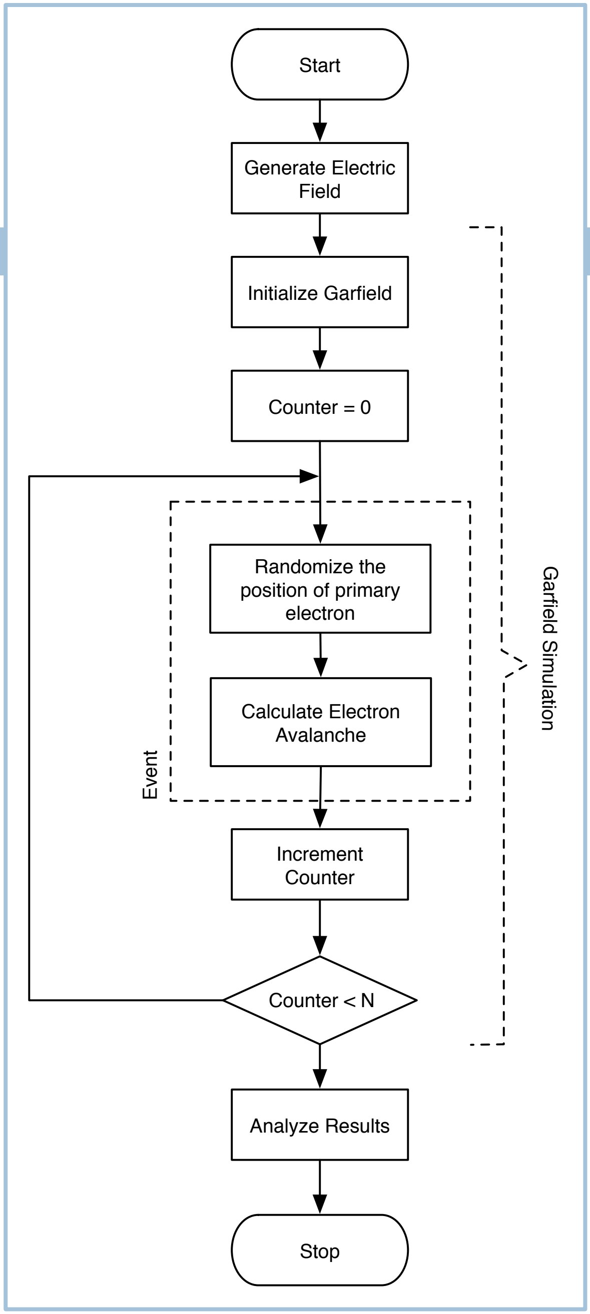
Event-level parallelism

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



Distributed memory architecture

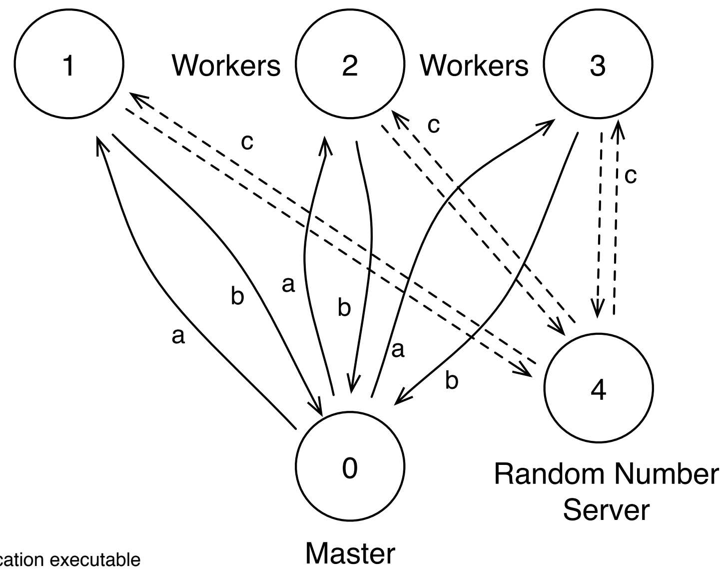
- Workflow of the serial simulation.

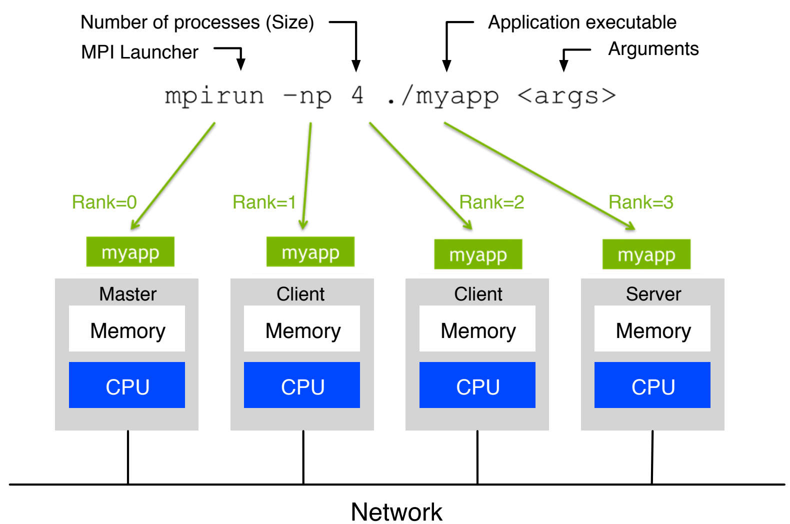


Serial simulation workflow

Parallel Garfield (pGarfield++)

- Based on MPI
- Architecture of the pGarfield++
- Random number generation
- Master distributes the workload and gathers the results back.

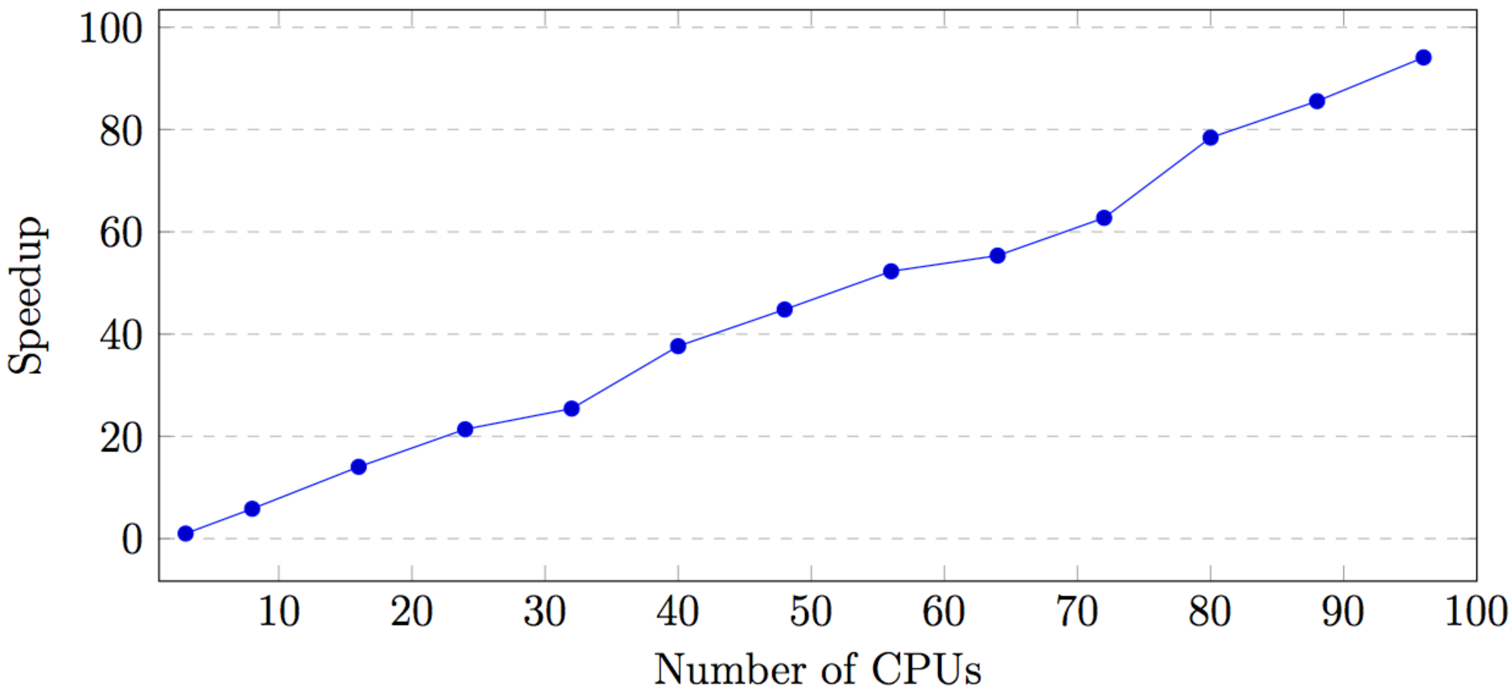




Performance Results: Speedup

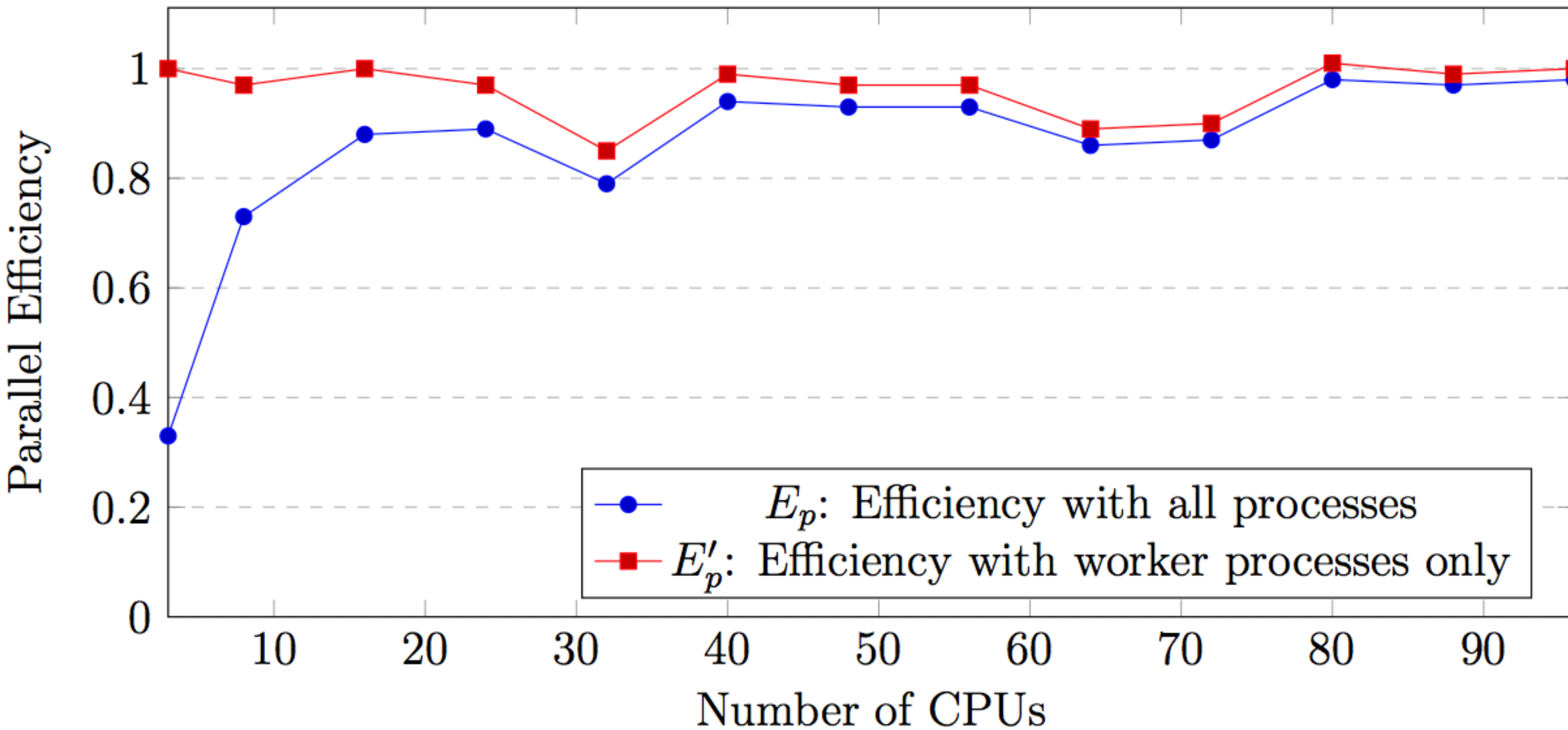
Performance was evaluated on the HPC cluster at Texas A&M University at Qatar. The HPC cluster (named RAAD) is a 42+ TFLOP, 2208-core Intel Xeon system.

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

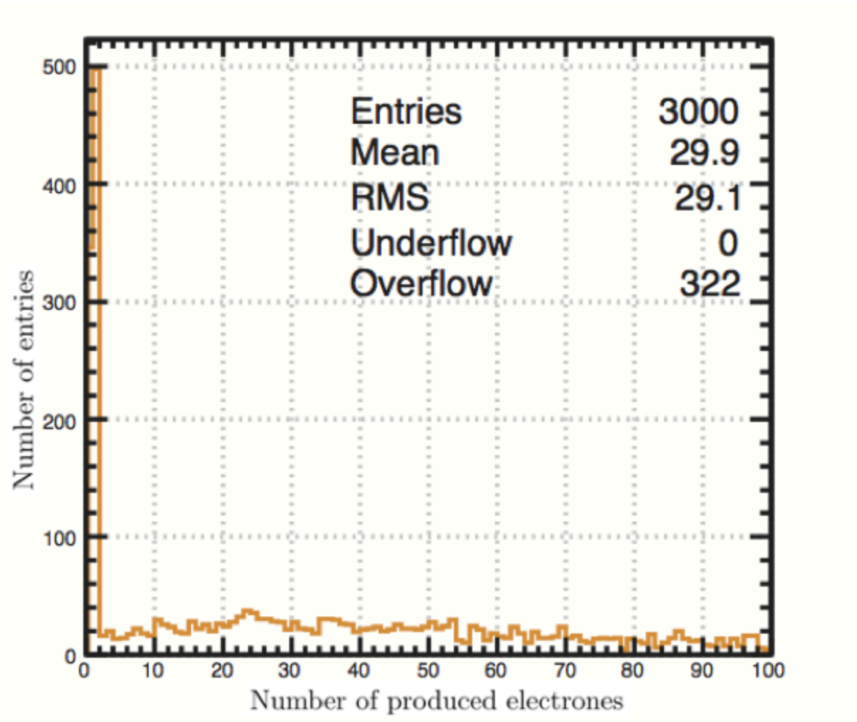


Performance Results: Parallel Efficiency

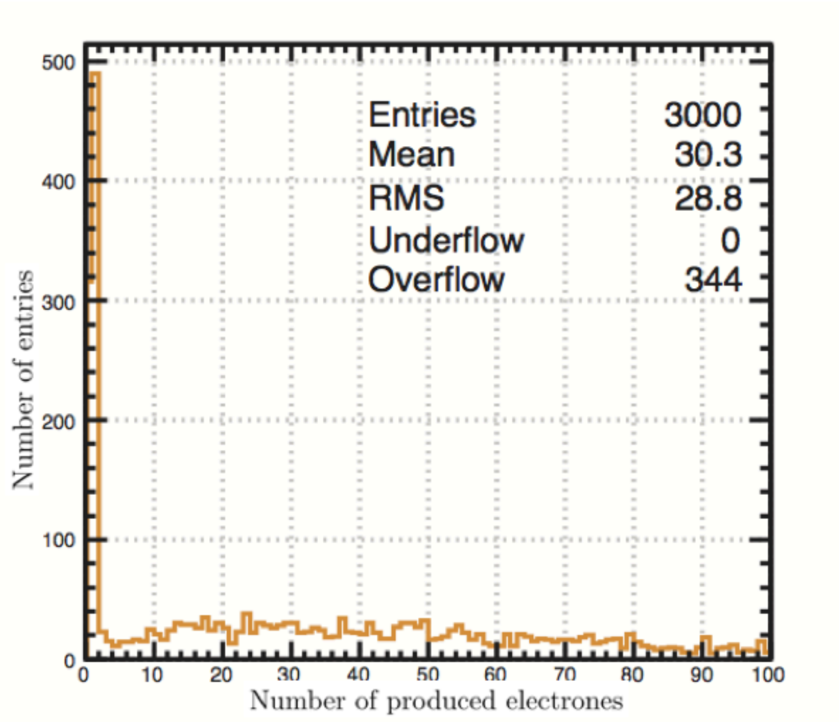
$$Efficiency = E_p = \frac{T_s}{p * T_p}$$



Correctness of the Parallel Simulation



(a) Histogram by the serial simulation



(b) Histogram by the parallel simulation

Summary

- Event-based parallelization completed
- Element Search optimization in progress (up to 26 times faster) and expect to increase

Next step:

- Improve search optimization
- Work on the track-based parallelization