Integration of CUDA GPU code in Garfield++

2nd DRD1 Collaboration Meeting

Tom Neep
University of Birmingham

Kostas Nikolopoulos
University of Birmingham
University of Hamburg

Mark Slater
University of Birmingham

2024-06-24

Introduction

- We are interested in simulating Townsend avalanches using `Garfield++`, specifically using `AvalancheMicroscopic`
- In the past we have found that simulating some high-gain detectors can take a very long time (tracking $10^5, 10^6, \ldots$ electrons)
- As the tracking of each electron in the avalanche is performed independently, this is a good (not perfect) scenario for performing in parallel
- GPUs are excellent at this and are becoming more common in academic environments (your institute likely has GPUs you can access if not then CERN does)
- In the past few years we have been working on adding CUDA (NVIDIA GPU) support to `Garfield++`
- Our code has now been added to the master branch of `Garfield++` (!398)

Updates

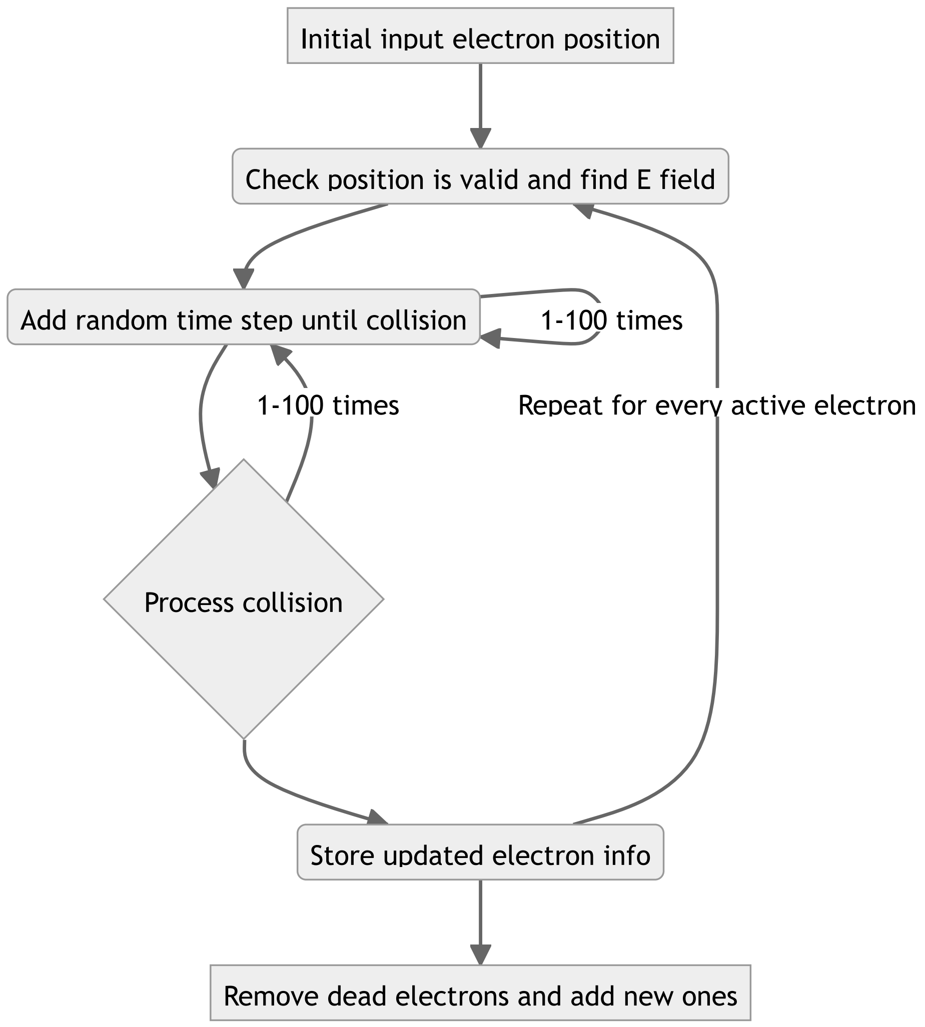
- Some of you may have been in my similar talk from RD51 meeting this time last year
- I will not cover some of the “work in progress” details that I discussed then, but please look back at the slides or ask questions if you are interested
- A lot of discussion in that talk about how we ensured that we get the same results on the CPU and GPU

GPUs

	CPU	GPU
Function	Generalized component that handles main processing functions of a server	Specialized component that excels at parallel computing
Processing	Designed for serial instruction processing	Designed for parallel instruction processing
Design	Fewer, more powerful cores	More cores than CPUs, but less powerful than CPU cores
Best suited for	General purpose computing applications	High-performance computing applications

Taken from https://aws.amazon.com/compare/the-difference-between-gpus-cpus/

AvalancheMicroscopic


Initial input electron position
Check position is valid and find E field
Add random time step until collision
1-100 times
1-100 times
Repeat for every active electron
Process collision
Store updated electron info
Remove dead electrons and add new ones

- A rough representation of a single iteration of `AvalancheMicroscopic`
- This is essentially the loop that we want to run on the GPU

UNIVERSITY OF BIRMINGHAM

Changes to your code

To install Garfield++ with GPU support, follow the instructions adding the USEGPU option when calling cmake

```
cmake -DUSEGPU=ON <src dir>
make -j 8
make install
```

and then change your code to run the avalanche on your GPU

```
AvalancheMicroscopic aval;
aval.SetSensor(&sensor);
aval.SetRunModeOptions(MPRunMode::GPUExclusive, 0);
```

You can run your code as normal and then retrieve the results with the GetNumberOfElectronEndpointsGPU and GetElectronEndpointGPU methods

```
unsigned int endpoints = aval.GetNumberOfElectronEndpointsGPU();
double xe1, ye1, ze1, te1, e1, xe2, ye2, ze2, te2, e2;
int status;
for (unsigned int i=0; i<endpoints; ++i) {
  aval.GetElectronEndpointGPU(
        i,
        xe1, ye1, ze1, te1, e1,
        xe2, ye2, ze2, te2, e2,
        status);
}
```

Changes to Garfield++

- We had to make several changes to the core `Garfield++` code
- Changes largely involve making GPU versions of classes
- Preprocessor macros used to “mark” code for GPU or CPU

```
❯ git diff --numstat master -- Source/ Include/Garfield/ | sort -nr
256 24  Source/ComponentFieldMap.cc
255 59  Source/AvalancheMicroscopic.cc
159 13  Include/Garfield/ComponentFieldMap.hh
138 14  Source/MediumMagboltz.cc
97  14  Include/Garfield/Medium.hh
92  0   Include/Garfield/AvalancheMicroscopic.hh
88  28  Include/Garfield/TetrahedralTree.hh
87  8   Include/Garfield/MediumMagboltz.hh
86  8   Include/Garfield/Component.hh
63  6   Include/Garfield/Sensor.hh
47  6   Source/TetrahedralTree.cc
34  6   Source/Sensor.cc
18  1   Include/Garfield/MediumGas.hh
15  0   Include/Garfield/MultiProcessInterface.hh
12  1   Include/Garfield/ComponentAnsys123.hh
8   0   Source/ComponentAnsys123.cc
3   0   Source/MediumGas.cc
3   0   Source/Medium.cc
3   0   Source/Component.cc
3   0   Include/Garfield/RandomEngineRoot.hh
3   0   Include/Garfield/RandomEngine.hh
```

Example changes required: arrays

ComponentFieldMap.cc

```
#ifdef __GPUCOMPILE__
__device__ void ComponentGPU::Jacobian13(
  const double xn[10],
  const double yn[10],
  const double zn[10],
  const double fourt0, const double fourt1,
  const double fourt2, const double fourt3,
  double& det, double jac[4][4])
#else
void ComponentFieldMap::Jacobian13(
    const std::array<double, 10>& xn,
    const std::array<double, 10>& yn,
    const std::array<double, 10>& zn,
    const double fourt0, const double fourt1,
    const double fourt2, const double fourt3,
    double& det, double jac[4][4])
#endif
```

- As an example, here is the signature of the `Jacobian13` method
- In this case, we simply need to replace `std::array` with C-style arrays

> The CUDA standard library has advanced since we started this work, and it may now be possible to use `cuda::std::array` to simplify these kind of changes in the future

Example changes required: vectors

ComponentFieldMap.hh

```
// Bounding boxes of the elements.
#ifdef __GPUCOMPILE__
GPUFLOAT** m_bbMin = nullptr;
GPUFLOAT** m_bbMax = nullptr;
#else
std::vector<std::array<double, 3> > m_bbMin;
std::vector<std::array<double, 3> > m_bbMax;
#endif
```

- Likewise we need to replace `std::vector` with dynamically allocated arrays (`cudaMallocManaged`)
- In this case we'd allocate the memory when the GPU version of the class is created and copy from the vector on the CPU

Benchmarking setup

- To benchmark the code we have adapted a model from `Garfield/Examples/Gem`
- This allows us to use an existing electric field map without needing to create a new one
- This example gives a low, unrealistic gain (≈ 4) but serves to demonstrate the performance of the CPU versus the GPU
- To test the performance, we start N electrons at the same position at the entrance of the GEM and see how long it takes to simulate the whole avalanche

Baskerville

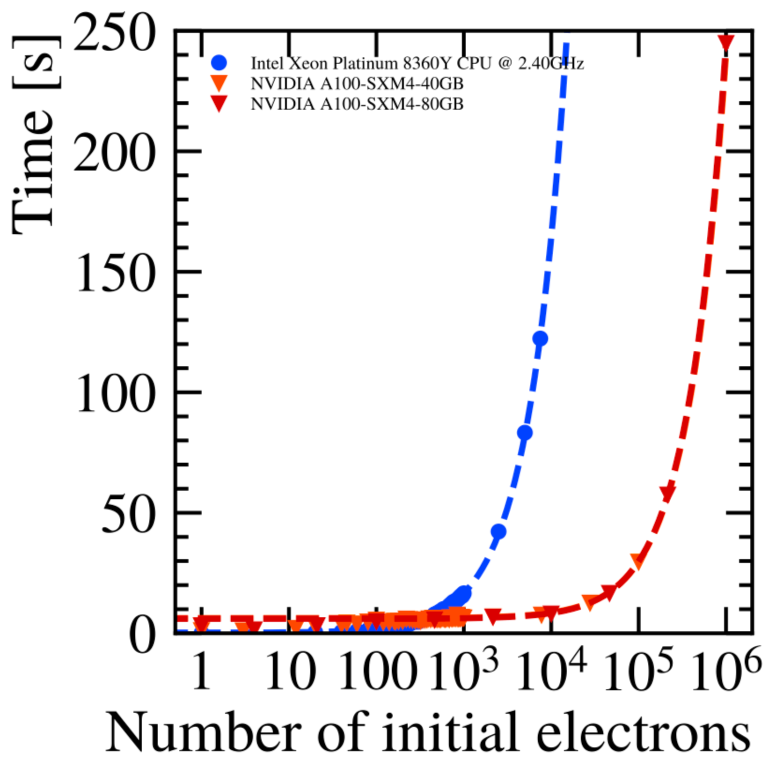
- For this presentation benchmarks were performed on the Baskerville HPC at the University of Birmingham
- 52 liquid-cooled compute trays, each two 36 core Intel Xeon Platinum 8360Y CPUs and four NVIDIA A100 GPUs (meshed with NVIDIA NVLINK)
- Currently 445th on the top500.org list

You do not need a supercomputer to benefit from this work!

Baskerville is funded by the EPSRC and UKRI through the World Class Labs scheme (EP/T022221/1) and the Digital Research Infrastructure programme (EP/W032244/1).

Benchmark results

- Run different number of electrons and fit a straight line to the timings
- GPUs up to 70× faster than CPUs for a large number of initial electrons
- Latest Nvidia GPUs have >2× more cores as A100 - would be interesting to test!



UNIVERSITY OF BIRMINGHAM

Running at CERN

1. `ssh` to `lxplus-gpu.cern.ch` (likely to get a NVIDIA Tesla T4 in my experience)
2. Run an HTCondor job (possibly interactively)

- Run the following command on lxplus to see the GPUs available:

```
❯ condor_status -constraint '!isUndefined(DetectedGPUs)' \
-compact \
-af GPUs_DeviceName TotalGPUs | sort | uniq -c
```

```
     31 NVIDIA A100-PCIE-40GB 1
      1 NVIDIA A100-PCIE-40GB 4
      1 Tesla P100-SXM2-16GB 1
      5 Tesla V100-PCIE-32GB 1
      1 Tesla V100-PCIE-32GB 4
     19 Tesla V100S-PCIE-32GB 1
```

Future work

- Our implementation of `AvalancheMicroscopic` is a starting point and there are many features in `Garfield++` that still don't work on the GPU e.g.
  - Calculating induced signals, which will likely have some *performance impact*
  - Using field maps not created by `Ansys` (should be fairly straight forward to add)
  - Some scope for tidying the user interface
  - Work on generic multithreading
  - **Isabella Oceano** from the University of Hamburg will be investigating some of these issues
- Additionally, there are some technical improvements that could be made:
  - As mentioned, using more of the CUDA standard library might further minimise changes required to CPU code to work on the GPU
  - It should be possible to start an avalanche on the CPU and then switch to the GPU when it reaches a certain size
  - Using multiple GPUs? Using non-Nvidia GPUs?
- We need to identify the features people need to perform their research effectively.

Summary

- You can now run on your Nvidia GPU by cloning the `Garfield++` master branch
- `AvalancheMicroscopic` has been implemented, with caveats
- Big speed ups for large avalanches
- Please try running your model on a GPU and see what problems (if any) you run into
- Thank you to **Mark Slater** for the initial GPU implementation and **Heinrich** for reviewing the merge request

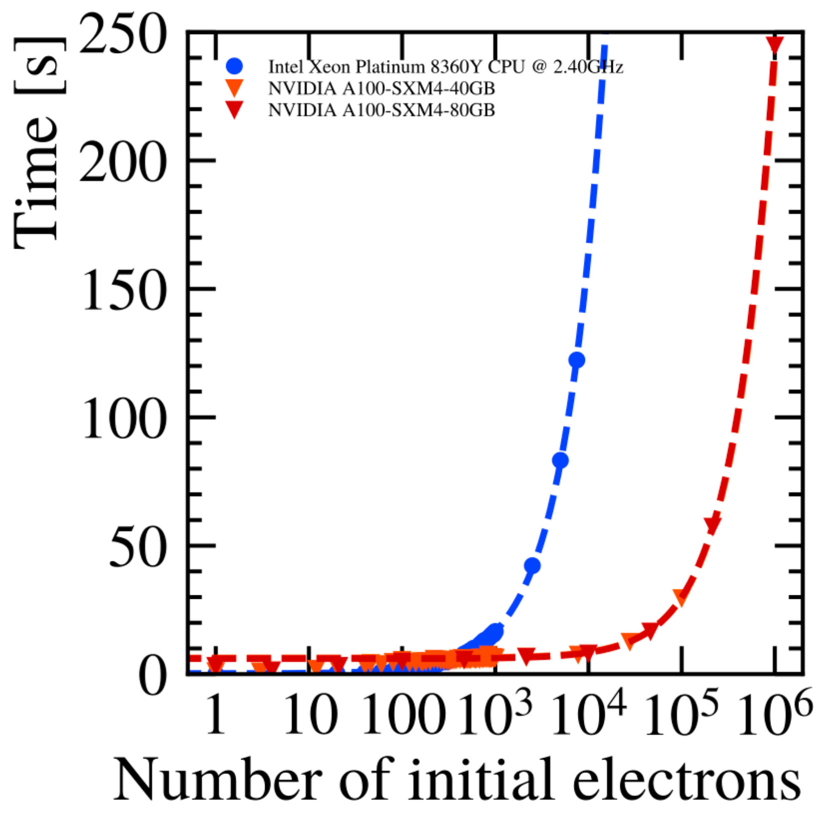
> We a preparing a paper which describes the work presented today

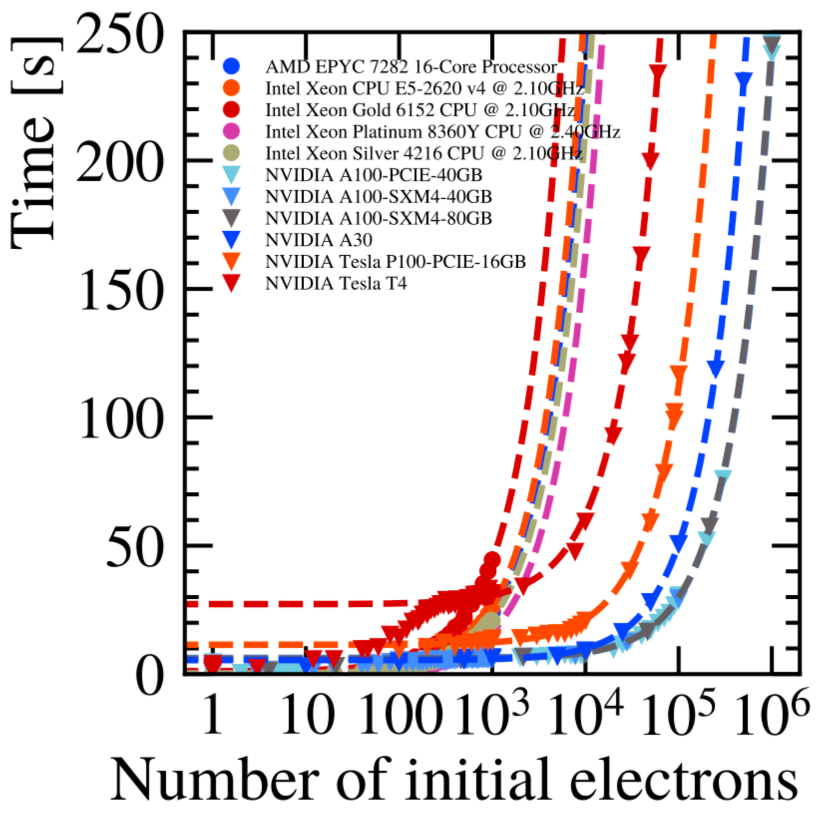
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no 101026519 (GaGARin).

Backup

Benchmark results

- Run different number of electrons and fit a straight line to the timings
- GPUs up to 70× faster than CPUs for a large number of initial electrons


Intel Xeon Platinum 8360Y CPU @ 2.40GHz
NVIDIA A100-SXM4-40GB
NVIDIA A100-SXM4-80GB
Time [s]
10³
10⁴
10⁵
10⁶
Number of initial electrons


AMD EPYC 7282 16-Core Processor
Intel Xeon CPU E5-2620 v4 @ 2.10GHz
Intel Xeon Gold 6152 CPU @ 2.10GHz
Intel Xeon Platinum 8360Y CPU @ 2.40GHz
Intel Xeon Silver 4216 CPU @ 2.10GHz
NVIDIA A100-PCIE-40GB
NVIDIA A100-SXM4-40GB
NVIDIA A100-SXM4-80GB
NVIDIA A30
NVIDIA Tesla P100-PCIE-16GB
NVIDIA Tesla T4
Time [s]
10³
10⁴
10⁵
10⁶
Number of initial electrons