Running GEANT4 Functions on a GPU

Discussion of Results

Stuart Douglas – dougls2
Rob Gorrie – gorrierw
Matthew Pagnan – pagnanmm
Victor Reginato – reginavp

McMaster University

April 14, 2016

Overview

1 Introduction
- Brief Project Overview
- Explanation of Terms
- Scope
- Purpose

2 Features
- Easily Enable/Disable GPU Acceleration
- Impl. 1: Existing Module in GPU Memory
- Impl. 2: Add New GPU-Accelerated Functions to Interface
- Accuracy / Testing

3 Conclusion
- Summary of Results
- Recommendations

Brief Project Overview

Take an existing particle simulation toolkit - GEANT4 - and have some functions run on a GPU device to improve performance.

What is GEANT4

- Geant4 is a toolkit that is meant to simulate the passage of particles through matter.
- It has been developed over the years through collaborative effort of many different institutions and individuals.
- Geant4's diverse particle simulation library has a wide variety of applications including
  - High energy physics simulations
  - Space and radiation simulations
  - Medical physics simulations

What is GP-GPU

- General purpose graphic processing unit computing is a re-purposing of graphics hardware
- Allows GPUs to perform computations that would typically be computed on the CPU
- If a particular problem is well suited to parallelization, GPGPU computing can greatly increase performance

Scope

- Make current CPU functions available for use on GPU
  - Add appropriate prefixes to function definitions
  - Make use of multiple parallel threads to execute each function
- Ensure correctness of each GPU available function by matching results to the corresponding CPU function
- Compare performance of GPU available functions to CPU functions

Purpose

- Determine if target functions are suitable to parallelization
- Increase performance of functions when run on GPU
- Decrease time required to run simulations involving ported functions

Easily Enable/Disable GPU Acceleration

- Old projects can be run using GPU acceleration with out having to change anything
- No new functions to learn
- GPU acceleration can be easily toggled using a flag during configuration and installation

Accelerating Module on GPU

Why G4ParticleHPVector

- Identified as starting point by relevant stakeholders
  - Used heavily in simulations run by stakeholders
- On the surface seems, suitable to parallelization
  - Uses large list data structure
  - Performs calculations on structure
- Loads in large amounts of data to a list structure

Two Implementations

- Run everything on the GPU
- Only select functions run on GPU

Completely on GPU

- The vector is stored exclusively on the GPU
+ Do not have to maintain a copy of the vector on the CPU
+ Do not have to maintain the hashed vector
+ Reduces how much is being copied to the GPU
– All functions are run on the GPU

Implementation – Times

Implementation – GetXSec

Implementation – SampleLin

Performance Results Summary

Performance Results – `Times`

- Multiplies each point in vector by factor

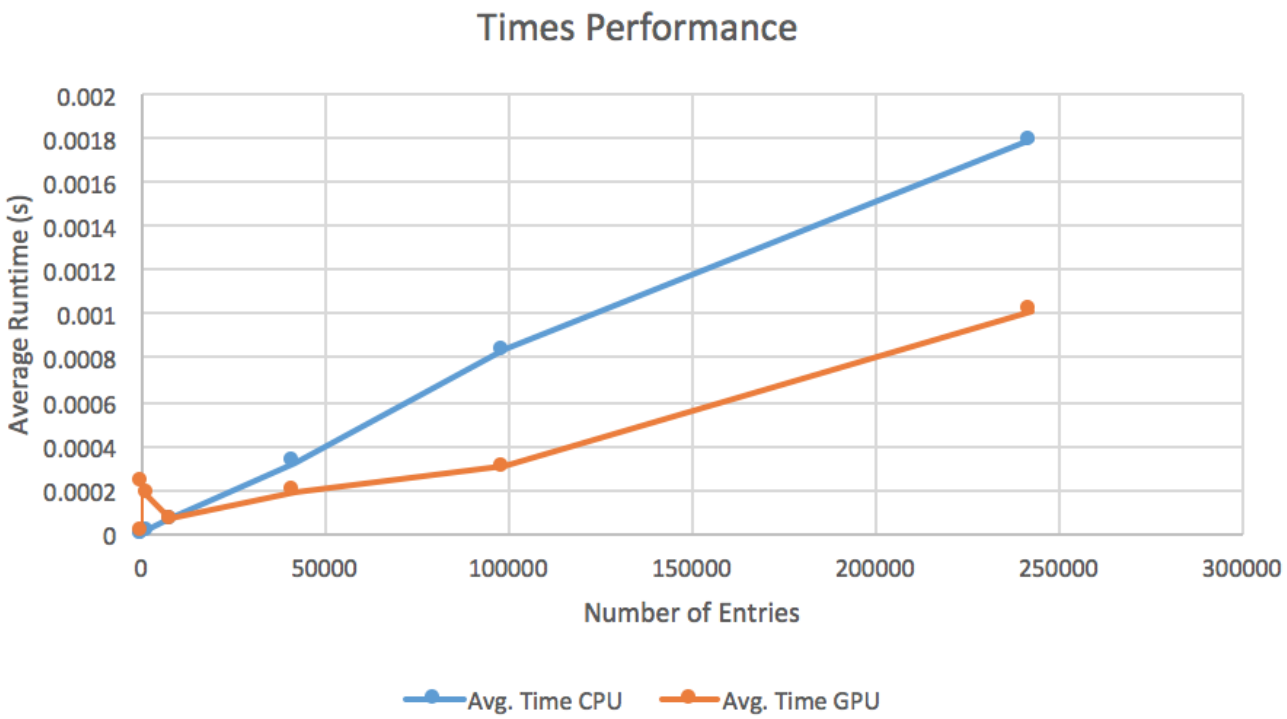

Figure: Runtime vs. Number of Data Points – `Times`

Performance Results – `GetXSec`

Performance Results – `SampleLin`

Performance Results – System Tests

Performance Discussion

Impl. 2: Add New GPU-Accelerated Functions to Interface

+ Only functions that run faster on the GPU are implemented
+ Not forced to run functions that run slowly on GPU
– Will have to maintain two copies of the vector
– More copying the vector to and from the GPU

Implementation – `Times`

Performance Results Summary

- Most functions slower on GPU until ~10,000 entries
- Most *commonly-used* functions significantly slower on GPU
  - Lots of data accesses
- Many problems in vector class not well-suited to parallelism

Performance Results – `GetXSeclList`

Performance Results – System Tests

Performance Discussion

Accuracy

Testing

Summary of Results

Recommendations