Geant4 GPU Port:

User Manual

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

Version 0
February 29, 2016

Contents

1 Revision History

All major edits to this document will be recorded in the table below.

Table 1: Revision History

Description of Changes	Author	Date
Initial draft of document	Matt	2016-02-26

2 List of Figures

Table #	Title

3 Definitions and Acronyms

Table 2: Definitions and Acronyms

Term	Description
Geant4	open-source software toolkit used to simulate the passage of particles through matter
Geant4-GPU	Geant4 with some computations running on the GPU
G4-STORK	(Geant4 STOchastic Reactor Kinetics), fork of Geant4 developed by McMaster's Engineering Physics department to simulate McMaster's nuclear reactor
GPU	graphics processing unit, well-suited to parallel computing tasks
CPU	computer processing unit, general computer processor well-suited to serial tasks
CUDA	parallel computing architecture for general purpose programming on GPU, developed by NVIDIA
RHEL	Red Hat Enterprise Linux Server
OS X	Operating system developed by Apple

4 Introduction

4.1 Purpose

4.2 Scope

4.3 Background

4.4 Document Overview

This document goes over how to install and run Geant4-GPU. As well as software and hardware required for Geant4-GPU to be installed and run. Following this are detailed step-by-step instructions on how to install the software. Following details on installation is a section on creating and running simulations using the CUDA functionality introduced in Geant4-GPU. A section detailing high level details about how the CUDA code was developed and integrated follows, guiding future developers and users on how to integrate CUDA into other parts of Geant4. A troubleshooting section is included as well as a FAQ section covering many common questions related to the project. Finally, the appendix includes conclusions about the state of the project and a brief description of the future of the project.

5 Legal Information

6 Installation

6.1 Supported Operating Systems

Geant4-GPU has been tested on OS X 10.11 and Red Hat Enterprise Linux Server release 6.7 (Santiago). Other versions of these operating systems are likely to work, but compatibility is not guaranteed.

To check what version of OS X you have, perform the following actions:

1. Click on the apple icon in the top left-hand corner of your screen.
2. Click on About This Mac.
3. Check to ensure that the version of your OS is 10.11 (El Capitan) (earlier versions may work, but are not guaranteed).

To check your version of RHEL, perform the following actions:

1. Open a shell.
2. Type in the following command `cat /etc/redhat-release`

3. Check to ensure that the version of RHEL installed is 6.7 (Santiago) (earlier versions may work, but are not guaranteed).

6.2 Recommended Knowledge

Geant4-GPU is a scientific product for use in research. There is a large learning curve to using Geant4, and it is recommended that the user has familiarity with the following.

- Comfortable with the Unix command line. This includes navigating the file system, executing make files, and installing software through the command line.
- Some knowledge of programming concepts and C++. Users develop their own simulations in C++ based off of the Geant4 toolkit, and as such some programming experience is recommended.
- Understanding of physics concepts. The user is expected to be knowledgeable in their domain, as Geant4 requires an understanding of the physics processes in the simulation.

6.3 Required Hardware

In order to successfully install and run Geant4 with GPU parallelism, you must first check that the workstation that you are installing on has a NVIDIA GPU with Compute Capability 3.0 or higher.

To check if the GPU on your current workstation supports CUDA with a Compute Capability of 3.0 or higher, perform the following actions:

1. Determine the graphics card that is currently installed (RHEL)
   (a) Open a shell.
   (b) Enter the following command: `lspci | grep -i nvidia`
   (c) You should see output similar to the following:
   `01:00.0 VGA compatible controller: NVIDIA Corporation GK104GL [Quadro K5000] (rev a1)`
   (d) Where [`Quadro K5000`] is the name of your NVIDIA Graphics Card.
2. Determine the graphics card that is currently installed (OS X)
   (a) Click on the apple icon in the top left-hand corner of your screen.
   (b) Click on About This Mac.
   (c) Click on More Info...

(d) Under Contents click Hardware to view it's options.

(e) Click on Graphics/Displays.

(f) You should see the name of your card at the top of the right view-pane.

3. Visit `https://developer.nvidia.com/cuda-gpus` and find your GPU to determine the Compute Capability of your card.

4. Check to ensure that you have a NVIDIA GPU with a Compute Capability of at least 3.0.

5. If you can't find the currently installed GPU on the list, it is not CUDA compatible.

6.4 Required Software

In order to install GEANT4-GPU you must have the following software installed:

- GCC version 4.8 (RHEL) or Clang version 6.0 (OS X). Note that McMaster's servers have GCC version 4.9, and an extra command is required to compile, as documented in section 6.5.2.
- CUDA Toolkit v6.5 (other versions may work, however 6.5 is the only one that is tested)
- CMake 2.8 or higher

6.4.1 Checking GCC Version (RHEL)

1. Open a shell.
2. Enter `gcc -v` into the command line.
3. You will see output that looks like: `gcc version 4.9.2 20150212 (Red Hat 4.9.2-6) (GCC)`
4. Ensure that the version number is 4.8

6.4.2 Checking Clang Version (OS X)

1. Open a shell.
2. Enter `clang --version` into the command line.
3. You will see output similar to:
```
Apple LLVM version 6.0 (clang-600.0.51) (based on LLVM 3.5svn)
Target:  x86_64-apple-darwin13.3.0
Thread model:  posix
```
4. Ensure that your version of Clang is `clang-600.x.xx`

6.4.3 CUDA Toolkit

If you don't already have the CUDA Toolkit installed, visit `https://developer.nvidia.com/cuda-toolkit-65` and follow the installation instructions for your OS.

After installing, check to see that CUDA is up and running. For either RHEL or OS X:

1. Open a shell.
2. Enter the following command "nvcc –version"
3. Output should be similar to:
```
nvcc:  NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2014 NVIDIA Corporation
Built on Thu_Jul_17_21:41:27_CDT_2014
Cuda compilation tools, release 6.5, V6.5.12
```

6.4.4 CMake

If you don't already have CMake (version 2.8 or higher) you can install it from: `https://cmake.org/download/`. After installing cmake, check to see that it is up and running. For either RHEL or OS X 10.11:

1. Open a shell.
2. Enter the following command: `cmake --version`.
3. Output should be similar to:
```
cmake version 3.4.0
CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

6.5 Installation Instructions

Included in the repository, there is a README.md file that breaks down the installation steps, including special instructions on how to install this on McMaster's servers. That document is also covered in this section of the user guide.

6.5.1 Installing Geant4-GPU

1. Either clone the repository or download it from `https://github.com/studouglas/GEANT4-GPU`
2. Open a shell

3. Navigate in the shell to the downloaded directory, and run the following command: `mkdir geant4.10.02-build geant4.10.02-install`

4. Navigate to the `geant4.10.02-build` directory you just created.

5. Run the following command to start the build process:
   `cmake -DGEANT4_INSTALL_DATA=ON -DGEANT4_ENABLE_CUDA=ON -DGEANT4_USE_SYSTEM_EXPAT=OFF -DCMAKE_INSTALL_PREFIX= /path/to/GEANT4-GPU/geant4.10.02-install /path/to/GEANT4-GPU/geant4.10.02`

   - `-DGEANT4_INSTALL_DATA=ON` – downloads and installs the datasets required by Geant4. This is only done once, and the argument may be ommited in subsequent Cmake commands.
   - `-DGEANT4_ENABLE_CUDA=ON` – setting this to `ON` will use your CUDA compiler to compile the CUDA code from the project and link it to the existing C++ code. Setting it to `OFF` (the default) will build Geant4 without any GPU functionality, and without requiring the CUDA compiler or a NVIDIA graphics card.
   - `-DGEANT4_USE_SYSTEM_EXPAT=OFF` – this simplifies the installation by using a version of the Expat library that is bundled with Geant4, rather than one that may or may not already be installed on the system.
   - `-DCMAKE_INSTALL_PREFIX=...` – sets the directory to install all the compiled Geant4 libraries, datasets, and examples.

   Note that running the above command for the first tame can take several hours depending on your internet connection, as it must download large datasets required by Geant4.

6. Once the previous step is completed, all the makefiles required to build Geant4-GPU are generated, all dependencies are installed, and all that remains is to actually compile Geant4.

   In your shell navigate to the `geant4.10.02-build` directory created in step 3 and run the following command: `make install`

   This step can take several hours depending on the speed of your computer.

7. Geant4-GPU is now installed and ready to be used with your simulations. For more information on creating simulations, see section 7.2. If you encountered any errors, please see section 9.

6.5.2 Installing on McMaster's GPU Servers

If installing Geant4-GPU on McMaster's GPU servers, the instructions above can be followed with minor differences:

- The default gcc compiler on the servers is too old to build Geant4, so add the following command to your `.bash_profile` to use a newer version:
  `. /opt/rh/devtoolset-3/enable`
- In step 5, add the following flag to the CMake command:
  `-DCUDA_HOST_COMPILER=/usr/bin/g++`. This compiles CUDA with the older version of gcc that's installed on the server, as the latest version is not supported.

7 Execution

Geant4 is not an executable program in the traditional sense. It is instead a large set of libraries, designed to work together and that give you, the user, a framework to develop simulations for your work. The development of such simulations is not within the scope of this manual, but it is well-documented with a thorough guide available at `http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/index.html`.

This section will instead outline how one would go about using (or not using) CUDA computations with an existing simulation. The `Hadr04` example that comes with Geant4 will be used as the simulation to demonstrate this.

7.1 Enabling/Disabling CUDA

The installation guide details how to build the Geant4 libraries that will be used by your simulation. The Cmake command from step 5 includes a flag that determines whether or not CUDA will be used – `-DGEANT4_ENABLE_CUDA=ON`. Changing this value to `OFF` will cause Geant4 to build without any CUDA features, and without requiring `nvcc`, the CUDA compiler. After step 5 finishes, users will have to rebuild Geant4 by executing the `make install` command from their `geant4.10.02-build` directory. To re-enable CUDA, simply change the value to `ON` and rebuild again.

7.2 Building the Simulation

After Geant4 has been built, building the simulation is very straightforward. First, navigate to your simulation's root folder (`geant4.10.02/examples/hadronic/Hadr04` in our case) in your terminal. Then, create a new directory `bin`, so your Hadr04 folder now has two folders, `bin` and `src`. Go into the `bin` folder and run `cmake ../src` followed by `make`. That's it!

7.3 Running the Simulation

There are two main methods of running a Geant4 simulation. The first is to simply run the executable, in our case by running `./Hadr04` in the Hadr04 directory. This launches an interactive Geant4 command prompt, and you can manipulate all aspects of the simulation from within it, including the types of particles, their number, and their energies. This also allows you to visualize the simulation using the `vis` commands.

The alternative is to create a macro file that contains a sequence of commands that would be manually given to the interactive prompt, making it much easier to run a given simulation more than once and without requiring user interaction. Both methods print the results of the simulation out to the terminal window, as well as the running time of the simulation. If you wish to save the results to a file, that can easily be done via redirection in Unix. For example, if you wish to save the results of your simulation using `myMacro.mac` to `results.txt`, simply run `./Hadr04 myMacro.mac > results.txt`.

8 Porting Other Geant4 Modules to CUDA

There is currently one class in Geant4-GPU that uses CUDA – G4ParticleHPVector. This class was originally chosen as it was thought to be well-suited to parallelism. Although certain functions within the class are, the majority of the time spent within the class is not in these functions, the main reason for the performance degradations when CUDA is enabled.

There is indeed the possibility that there are other classes within Geant4 that are better suited to parallelism. This section will outline the general methodology we used to port G4ParticleHPVector with the hopes that it may be of use to future developers or users.

8.1 Layout of Source Files

An important consideration during development was to keep the CUDA code separate from the main Geant4 codebase as much as possible. We created a folder named `cuda` in `geant4.10.02/source/externals` to contain all CUDA source code. The `cuda` folder contains two folders, `include` and `src`. All header files are in `include`, and the CUDA source files are in `src`.

8.2 CMake Changes

Geant4 uses CMake as its build system, and minor modifications need to be made to integrate with CUDA. This consists of adding a variable at the top level CMakeLists file to enable or disable CUDA, and creating a Geant4_Add_Library_Cuda macro

to Geant4MacroLibraryTargets. This uses `cuda_add_library` function included in Cmake. More details can be found in the project's detailed design document.

8.3 Interfacing with Existing Code

We used compiler directives within the default G4ParticleHPVector.cc file to make use of the CUDA code or to use the original implementation based on the flag passed to Geant4 during the build phase. If the flag is true, an object of type G4ParticleHPVector_CUDA is initialized and all function calls become one line calls to the corresponding function on the G4ParticleHPVector_CUDA object. For example, the `GetY` function looks like the following:

```
G4double GetY(int i) {
        #if GEANT4_ENABLE_CUDA
                return cudaVector->GetY();
        #else
                < existing code from G4ParticleHPVector.cc >
        #endif
}
```

8.4 Naming Conventions

It is important to use naming conventions to ensure clarity within a project that contains many possible ambiguities. The two main conventions we used were adding a "_CUDA" suffix to all source files within the `cuda` directory as well as to the class name of any classes implemented in CUDA. In addition to this, within all CUDA files there are two main types of pointer – those to GPU memory and those to CPU memory. To distinguish between these, all pointers to main memory are prepended with "h_" and pointers to GPU memory with a "d_".

9 Troubleshooting

9.1 Installation

- Ensure path names are correct in Cmake command
- Spaces in path names may cause issues on some systems
- The paths in step 5 of installing Geant4 paths must be absolute
- Newer versions of Clang (included with Xcode 7) have been know to cause problems. Download Xcode 6 and uninstall Xcode 7 if this is the case.

9.2 Running the Simulation

- Ensure that your graphics card meets the requirements detailed in section 6.3

9.3 FAQ

9.3.1 What is Geant4?

Many physics researchers use Geant4 to learn about how particles interact with a specific environment. It is a toolkit (i.e. set of libraries) that uses the Monte Carlo model, meaning each particle's properties are calculated independently according to certain probabilities. Users develop simulations based on these libraries.

9.3.2 Why will a GPU improve the performance?

GPU's contain a large amount of cores that can perform calculations much more quickly than a CPU if the problem is well-suited to parallelization. Geant4 runs relatively simple calculations on millions of particles, and each particle is completely independent of the others. This is exactly that sort of well-suited problem, and stands to see large performance gains.

9.3.3 Where can I find more information about Geant4?

For more information, check out the official website at `http://www.geant4.web.cern.ch/geant4/index.shtml`

9.3.4 Helpful Pages:

- Download page for Geant4 source code: `http://www.geant4.web.cern.ch/geant4/support/download.shtml`
- Getting Started guide: `http://www.geant4.web.cern.ch/geant4/support/gettingstarted.shtml`
- Installation Guide for Geant4 (does not include CUDA): `http://www.geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/InstallationGuide/html/index.html`

10 Appendix

10.1 Recommendations for Integration of Geant4 and CUDA

10.2 Future of Geant4-GPU