EM Physics on CUDA - Bremsstralung

Philippe Canal, Daniel Elvira, Dongwook Jang*, Soon Yung Jun, James Kowalkowski, Marc Paterno, Panagiotis Spentzouris

Fermilab

* Carnegie Mellon University

Contents

- Introduction
- Simplified strategy
- Implementation details
- Validation
- Conclusion

Introduction

- Extension to the transportation which was presented in the previous meeting by Soon Jun:
  https://indico.cern.ch/getFile.py/access?contribId=4&resId=0&materialId=slides&confId=189492
- Adding EM physics will increase computational intensity on GPU
- EM physics is a dominant physical interaction
- Multiple stepping will be possible on each thread if fully implemented
- Efficient memory management on both CPU and GPU side is critical

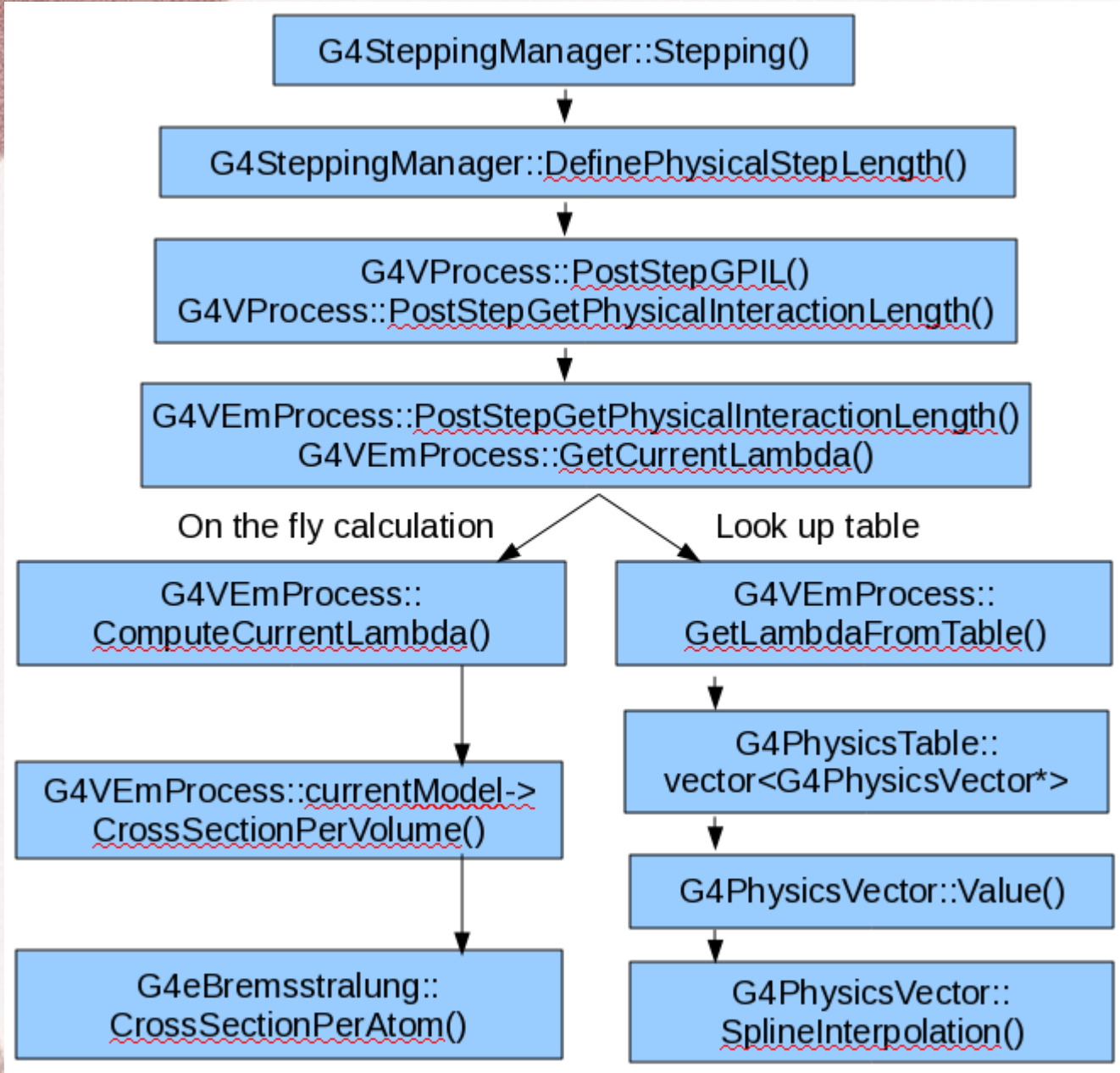
Simplified Strategy

- A cross section table is prepared on host side and copied to device side as a global memory.
- Track data (input) is prepared on host and copied to device as global memory and copied back to device containing results at the end of kernel execution
  - for efficiency, track data is overwritten with physical interaction length information
- Simple material (PbWO4)
- A physics model : bremsstralung for this study
  - PostStepGetPhysicalInteractionLength
  - PostStepDoIt : not done yet

Implementation Details

- A standalone package : no dependency on Geant4
  - This is only for testing purpose
  - At the time of plugging this into Geant4 in the future, there may be some dependencies.
- Convert c++ classes to c-struct
- C++ member functions to generic functions with passing object pointers as arguments.
- Convert std::vector to the fixed size array
- Follow Geant4 functional flow as similar as possible, but simplified by removing unused branches

Functional Flow

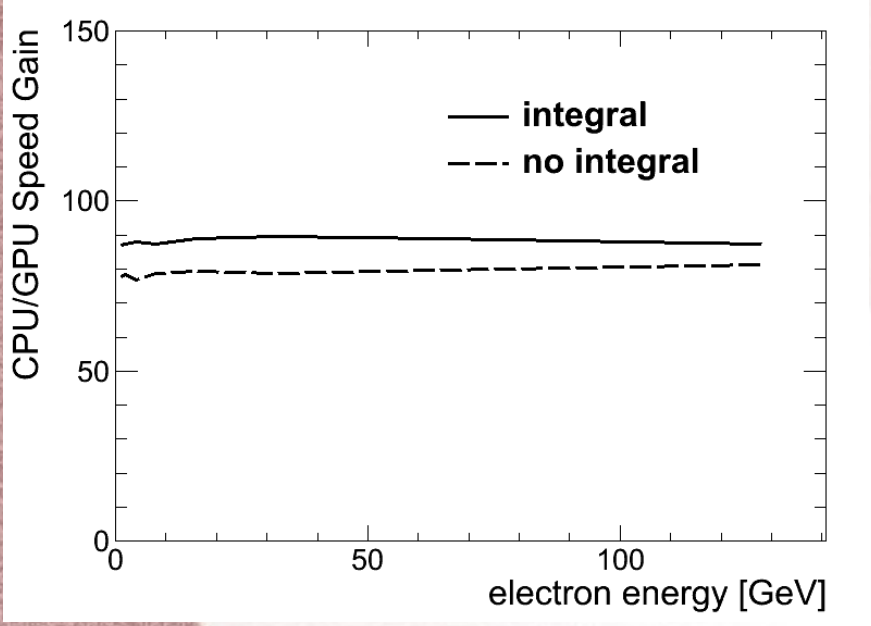


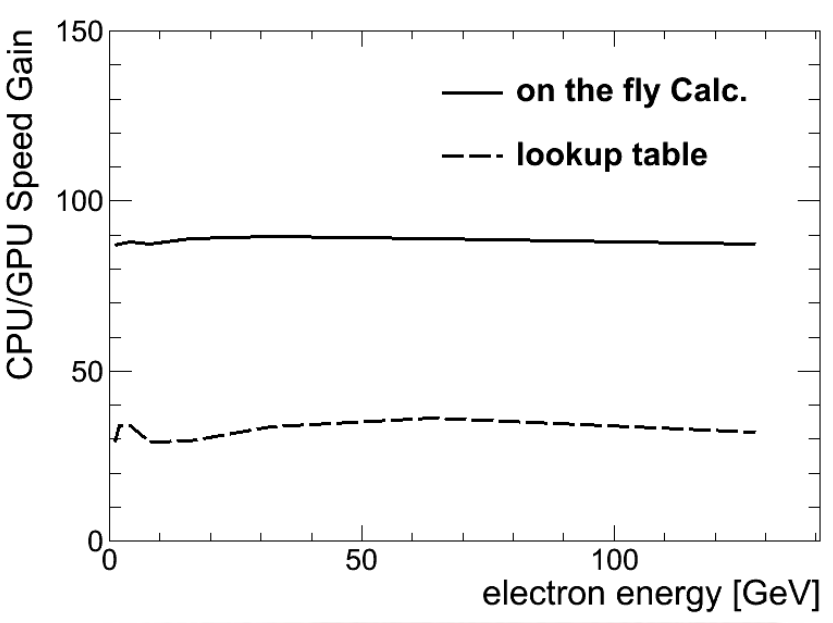
Validation

- Testing environment
  - CPU (host) : AMD Opteron 6136 2.4 GHz
  - GPU (device) : 1.15 GHz 448 cores (jobs are divided into 32 blocks with 128 threads, no optimization on blocks and threads numbers is done yet)
  - 100K electrons are prepared as an input
  - Detector material : PbWO4 (hardcoded)
  - CPU/GPU kernels are identical
  - One stepping is tested
- Definition of CPU/GPU time
  - CPU time : kernel execution on single CPU
  - GPU time : kernel execution on 448 cores + memory allocation on device + memory copying from host to device + memory copying from device to host

Validation

- Geant4 default is using integral and looking up tables
- Overall gain is about a factor of 30 for the default configuration





Numbers

- The actual time for electron 16 GeV case

	On the fly	Lookup table
CPU [ms]	1189.3	49.9
GPU [ms]	13.4	1.7

Conclusion

- We have observed some benefits by adding a small part of EM physics on GPU ~ a factor of 30
- Implementation is still limited and simplified compared to Geant4
- Efficient memory handling and optimal track dispatching is needed when implementation is in mature stage
- Will add energy loss parts for brem soon
- Will add more standard EM physics if brem implementation is done: very challenging