Prototyping Geant On GPU

Philippe Canal, Daniel Elvira
Soon Jun, James Kowalkowski
Marc Paterno, Panagiotis Spentzouris
Fermilab
Dongwook Jang
Carnegie Mellon University

Geant4 Collaboration Meeting, September, 2012

Overview

- Charge and Goals
- Prototypes
- Results and Conclusions

Note: This is a summary presentations, additional details are available in the two additional detailed presentations available from the meeting agenda.

Charge

- Develop and study the performance of various strategies and algorithms that will enable Geant4 to make efficient use of multiple computational threads
- Analyze the internal architecture of Geant4
- Profile and document performance and memory requirements for typical HEP applications
- Identify components that require re-engineering
- Begin developing prototypes of the new components

Specific Goals

- Investigate porting specific portion of Geant4 to GPU and answer the questions:
  - What is the performance?
  - What modifications does it imply?
  - How can it be integrated with general purpose code?

Specific Goals

- Understand
  - Geant4 code structure
  - Coding and Optimization on GPU (Tesla)
  - How the two can be matched
  - If the same style of modification benefits CPU code
- Provide Feedback to global re-engineering effort

How

- Bottom-up approach
  - Extract time consuming proportion of the code
  - Feed prototype with realistic data
    - Captured from running a full Geant4 example
- **Experimental software environment**: cmsExp
  - CMS geometry (GDML) and magnetic field map (2-dim grid of volume based field extracted from CMSSW)
  - Shooting 100 GeV Pions

What

- **Equation of motion in magnetic field (Stepper)**
  - Classical 4th Order Runge Kunta
  - Include simple magnetic field lookup
  - CUDA implementation and optimization
  - Trying out various techniques and memory layout
  - Tested texture memory and CUDA Streams
- **Geometric limit of step length**
  - Navigator and multilevel locator
  - Limited set of shapes
    - Original work by CERN summer student (O. Seisaski)
  - Simple detector similar in structure to the CMS central electromagnetic calorimeter

What

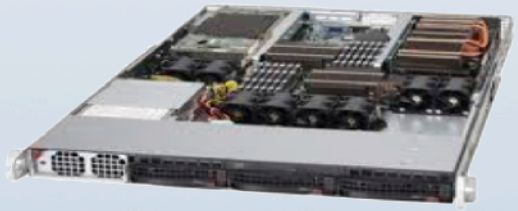
- **Particle Transportation (no physics)**
  - Combines stepper and geometry
  - Extended to particle propagation component
    - Focus on the magnetic field case
  - Support the transportation of neutral particles (photons) and charged particles without a magnetic field
  - Complete charged particle transportation under a magnetic field including the geometrically limited step control.

What

- **EM Physics on the GPU**
  - Tested both ***cross section table*** and on the fly calculation
  - Simple material (Stolzite, $PbWO_4$)
  - A physics model : bremsstrahlung for this study
    - PostStepGetPhysicalInteractionLength
    - PostStepDoIt : not done yet
- **Particle Transportation with EM Physics**
  - Combination of all the prototypes

Hardware

- **Host**: AMD Opteron Processor 6136
  - 4 CPU: 2.4 GHz, Processors: 8 cores
  - L1/L2/L3 Cache Size: 128/512/12288 (KB)
  - L3 Cache speed: 2400 MHz



- **Device**: NVIDIA Tesla M2070
  - GPU clock speed: 1.15 GHz
  - 14 Multiprocessors x 32 CUDA Cores: 448 CUDA cores
  - Memory: global 5.4 GB, constant 65 KB, shared 50KB
  - L2 Cache size: 786 KB
  - Maximum thread per block: 1024
  - Warp size: 32
  - CUDA Capability Major/Minor: 2.0

Performance Numbers

Ratio of Processing Time: CPU/GPU	Kernel (Computation) Only	Kernel + Data Transfer
Stepper	67	29
Geometric limit of step length	74	32
Particle Transportation (no physics)	33	27
EM Physics	88	9
Particle Transportation with EM Physics	34	30

Cost Comparison

- Est for NVIDIA Tesla M2070: $2500
- Est price of CPU (Opteron Processor 6136): $1000 -> One core $125
- Roughly a whole node consume half as much energy as a GPU card.
  - In some cases, this ratio can reach 1 or more (CPU uses as much as GPU)
- GPU Card 30 faster than one commodity Core
  - GPU Card 3.75 faster than one CPU (8 cores)
- GPU Card 20 x the price of one Core
  - GPU Card 2.5 x the price of one CPU
- Per unit of work (per gflops), the GPU costs:
  - 2/3 of the purchasing price of the CPU
  - ***At most 54% of the operating cost of the CPU (18% for well tuned code)***

Lesson Learned

- Gain greater than cost differential
- Optimization increase those gains
- Limiting factor: CUDA kernel setup and transfer speed
  - Requires limiting the frequency of data exchanges
- Texture memory
  - Help in truly random memory lookup
- CUDA Streams
  - Hide (most of) data transfer latency as long as computation time is greater or equal to transfer time.

What's next

- More complex magnetic field lookup (ATLAS?)
- Other equation of motion integrator (Nystrom)
- Extend the geometry set
- Explore implementation strategy for geometrical algorithm
- Optimize existing prototypes
- Test on other upcoming platforms (MIC / Knights Corners)