Celeritas: GPU detector simulation

Seth R Johnson

HPC methods for nuclear applications

Celeritas core team:
Philippe Canal, Stefano Tognini, Soon Yun Jun, Guilherme Lima, Amanda Lund, Vincent Pascuzzi, Paul Romano

Compute Accelerator Forum
29 June, 2022
Background
Celeritas overview

• **GPU-targeted** re-implementation of a *subset* of Geant4 physics leveraging both HEP physics community and HPC/GPU particle transport domain knowledge

• First code committed June 2020

• First DOE programmatic funding allocation: July 2022 🌈

• Short-term application: offloading EM tracks from Geant4 to GPU (*Acceleritas* bridge library)

• Long-term application: direct high-performance integration into LHC analysis workflows
High Performance Computing (HPC) in HEP

- High Luminosity upgrade means 10× higher sampling rate
- More detector data means more simulations needed
- Tens of millions of “equivalent 2006-era CPU hours” for analysis
- 20–25% is from full fidelity MC

MC simulation requirements: projection assumes 2× performance per watt GPU/CPU
GPUs now dominate calculation throughput on HPCs

- General Purpose Graphics Processing Units (GP-GPU)
  - Conceptualized in early ’00s
  - Very fast and power efficient for “graphics”-like applications
  - “Many-core”: massively multithreaded
- Programming models require much more care
  - Not good at flexible/dynamic operations
  - Ideally lots of operations per memory access

©NVIDIA Corporation, reproduced with permission
Challenges

• Execution: divergence and load balancing
  • GPUs want every thread doing the same thing
  • MC: every particle is doing something (somewhat) different

• Memory: data structures and access patterns
  • GPUs want direct, uniform, contiguous access
  • MC: hierarchy and indirection; random access
  • Memory allocation is a particular problem
Code design

• Core principles
  • Data-oriented programming
  • Object-oriented interfaces to data
  • Composition-based objects
  • Revisit legacy design/implementation choices

• Development workflow
  • Extensive unit testing in CPU execution space
  • Some unit testing and more integration testing on GPU
  • In-depth merge request review process
  • Continuous integration

Easily refactored for new architectures, data models, performance
Features
Memory model for hierarchical data

• Define data structures once
• Easily assemble data on CPU
• Data and execution on both CPU and GPU
• Single-line data transfer

Safe and effective framework for physicists to implement and test GPU-compatible physics
Multi-architecture portability

• Macro-decorated, header-only, (inline) C++ execution code *(no fancy CUDA)*

• Kernel “launcher” inline functions operate on a single thread's data

• Auto-generated CUDA/HIP, OpenMP, stdpar (NVIDIA collab)

*Requirement for universal DOE LCF usability*
VecGeom integration

GPU-traced rasterization of CMS 2018
ORANGE

Oak Ridge Advanced Nested Geometry Engine

- Designed for deeply nested reactor models
- Portable (CUDA/HIP) geometry implementation for testing
- Tracking based on CSG tree of surfaces comprising volumes
- Maximize run-time performance by preprocessing

Image credit: Steve Skutnik (ORNL)
**ORANGE surface-based tracking methodology**

**Celeritas geometry interface**

- **Primary initialization**
  - Initialize
- **Secondary initialization**
  - Fast-initialize
  - Move (to boundary or internal)
- **Along-step**
  - Find next step
  - Find safety
- **Boundary**
  - Cross boundary

<table>
<thead>
<tr>
<th>Position</th>
<th>Volume</th>
<th>Surface+Sense</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>Find step</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>Move internal</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>Move to bdy</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>Cross bdy</td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>Move internal</td>
<td>D</td>
<td>2</td>
</tr>
</tbody>
</table>

![Diagram of tracking methodology](image)
ORANGE surface/volume construction

ORANGE geometry definition
SCALE input
Triton reactor description
GDML/VecGeom

ORANGE construction object model
Shapes
Transforms
Cells
Universes
Arrays
Transformed surfaces
Simplified surfaces
CSG tree

CSG leaf
CPU Diagnostics
Metadata

GPU Runtime

Not yet available in Celeritas ORANGE
### Standard EM physics

<table>
<thead>
<tr>
<th>Particle</th>
<th>Process</th>
<th>Model(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>photon conversion</td>
<td>Bethe–Heitler</td>
</tr>
<tr>
<td></td>
<td>Compton scattering</td>
<td>Klein–Nishina</td>
</tr>
<tr>
<td></td>
<td>photoelectric effect</td>
<td>Livermore</td>
</tr>
<tr>
<td></td>
<td>Rayleigh scattering</td>
<td>Livermore</td>
</tr>
<tr>
<td>$e^\pm$</td>
<td>ionization</td>
<td>Møller–Bhabha</td>
</tr>
<tr>
<td></td>
<td>bremsstrahlung</td>
<td>Seltzer–Berger, relativistic</td>
</tr>
<tr>
<td></td>
<td>pair annihilation</td>
<td>EPlusGG</td>
</tr>
<tr>
<td></td>
<td>multiple scattering</td>
<td>Urban</td>
</tr>
<tr>
<td>$\mu^\pm$</td>
<td>muon bremsstrahlung</td>
<td>Muon Bremsstrahlung</td>
</tr>
</tbody>
</table>

- **Diagram:**
  - **Urban MSC comparison (in progress!)**
  - Graph shows comparison of different models for electron and gamma rays.
  - Axes: Steps per track (x-axis), Number of tracks (y-axis).
Transport loop and control flow

**Pre-step kernel**
- Track slot active?
  - Yes: Discrete action
  - No: No action
- Slowing down?
  - Yes: Range action
  - No: Discrete action
- Range and step limiters
- Fixed step action

**Along-step kernel**
- MSC step limiter
- Hit boundary?
  - Yes: Boundary action
- Stopped w/o rest?
  - Yes: Kill

**Discrete kernel**
- Integral XS Rejection
  - Yes: No interaction
  - Sample process
    - Model action

**Boundary kernel**
- Exited world?
  - Yes: Kill
Geant4 integration

- GDML file plus basic EM list option loads physics data
- *Acceleritas* bridges Celeritas directly to Geant4 run manager
- **In progress:** direct VecGeom geometry from in-memory Geant4
  - Currently: separate VGDMML call constructs VecGeom
  - Future: construct ORANGE representation automatically
- Not all Geant4 data is accessible via APIs
  - Seltzer–Berger data read from files specified in G4LEDATA
  - Some cross sections are constructed on-the-fly from models
New features for 0.1.0 🦄

- Polished library experience
  - Stable API for runtime setup and transport
  - Code documentation and manual
  - Separated core/ORANGE/celeritas layout
  - Integrates as installable library or as CMake project subdirectory

- Transport on realistic materials (sampling over elements)

- Transport in generalized magnetic field (provide “uniform” option)

- Multiple scattering (still undergoing validation)
Performance
Benchmark problem

- **TestEm3 — simplified calorimeter**
  - 50 alternating layers of Pb and lAr
  - 10,000 10 GeV electron primaries

- **Equivalent configurations of Celeritas/Geant4/AdePT**
  - No magnetic field
  - Disabled multiple scattering, energy loss fluctuations, Rayleigh scattering
  - Excludes initialization time

- **No spline interpolation in Celeritas**
  - ~3% performance penalty for Geant4 with spline
  - Compensate by using 8× cross section grid points: <2% slower
Initial performance results

• Per-node performance
• 1–2 batches of 6 simultaneous runs on Summit
  • CPU: multithreaded with 7 cores
  • GPU: one CPU core per GPU
• 40× faster with GPUs
  • Apples-to-apples: Celeritas CPU vs GPU
  • Similar order-of-magnitude improvement irrespective of code
  • 280 CPU core to GPU equivalence

Wall time per primary (ms)

<table>
<thead>
<tr>
<th></th>
<th>geo</th>
<th>arch</th>
<th>mean</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geant4 10.7.1</td>
<td>Geant4</td>
<td>CPU</td>
<td>2.9</td>
<td>0.1170</td>
</tr>
<tr>
<td>Celeritas 8d83ebab (29 Apr 2022)</td>
<td>ORANGE</td>
<td>CPU</td>
<td>2.09</td>
<td>0.0192</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>0.046</td>
<td>0.0012</td>
</tr>
<tr>
<td>VecGeom</td>
<td>CPU</td>
<td></td>
<td>1.95</td>
<td>0.0352</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>0.0627</td>
<td>0.0004</td>
</tr>
</tbody>
</table>

Number of primaries per run

<table>
<thead>
<tr>
<th></th>
<th>Geant4</th>
<th>Geant4</th>
<th>CPU</th>
<th>1E+04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celeritas</td>
<td>ORANGE</td>
<td>CPU</td>
<td>1E+03</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>1E+05</td>
<td></td>
</tr>
<tr>
<td>VecGeom</td>
<td>CPU</td>
<td></td>
<td>1E+03</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU</td>
<td>1E+05</td>
<td></td>
</tr>
</tbody>
</table>
Detailed timing

```plaintext
extend_from_primaries
while Tracks are alive do
  initialize_tracks
  pre_step
  along_step
  boundary
  discrete_select
  launch_models
  extend_from_secondaries
end while
```

- Copy primaries to device, create track initializers
- Create new tracks in empty slots
- Sample mean free path, calculate step limits
- Propagation, slowing down
- Cross a geometry boundary
- Discrete model selection
- Launch interaction kernels for applicable models
- Create track initializers from secondaries

![Timeline Diagram](image)

- initialize_tracks -
- pre_step -
- **along_step** -
- cross_boundary -
- discrete_select -
- launch_models -
- extend_from_secondaries -

Time [s]
Upcoming performance testing

- Multiple scattering
- Tracking in magnetic field
- *Acceleritas* multithreaded scaling
- Frontier single-node performance (AMD GPUs)
Celeritas is open for business!

• Reached a minimal level of feature completion to be useful
• Proven performance advantage (for test problems so far)

• Key areas of continuing work:
  • Physics validation (physics models, progression problems, experiment-specific)
  • Experiment integration (Acceleritas, or directly)
  • Performance experimentation (there’s a long list)
  • International collaboration (AdePT, VecGeom, ORANGE)