Celeritas: conclusion and challenges

Seth R Johnson

Celeritas core team:

Philippe Canal, Stefano Tognini, Tom Evans, Soon Yun Jun, Guilherme Lima, Amanda Lund, Vincent Pascuzzi, Paul Romano

HSF Detector Simulation on GPU Community Meeting
May 3–6, 2022

Summary

Performance

- Cautions
  - Single datapoint with simple feature set
  - Different code capabilities, RNG quality, implementation choices
- Results
  - ~280 GPU/CPU performance equivalence for Celeritas
  - Speedup from implementation choices alone is far from 2×!

Wall time per primary (ms)

	geo	arch	mean	σ
Geant4 10.7.1	Geant4	CPU	2.9	0.1170
AdePT 68508ef7 (sethrj/adept/summit, 2 May 2022)	VecGeom	GPU	0.0850	0.0005
Celeritas 8d83ebab (29 Apr 2022)	ORANGE	CPU	2.09	0.0192
		GPU	0.046	0.0012
	VecGeom	CPU	1.95	0.0352
		GPU	0.0627	0.0004

Implementation speedups (t_B / t_A – 1)

ORANGE / VecGeom (GPU)	35%
Celeritas (GPU, VecGeom) / AdePT	36%
Celeritas (CPU, VecGeom) / Geant4	50%

TestEM3 $^w/_o$ MSC or energy fluctuations single Summit node

Features

- Standard EM physics and VecGeom geometry
- Extensible, refactor-friendly code design
- Runtime problem configuration
- Imminent v0.1.0 release targeting:
  - Transport loop API for external code integration through Acceleritas
  - Runtime-configurable EM physics and magnetic field
  - ORANGE component availability

Future work and challenges

Physics validation and progression problems

- Verification, validation, performance for cross-code comparison
- Combinatorial features for debugging with experimental "control"
- Increasing complexity for immature or experimental codes

	Condition			
Tested feature	Physics models	Material	Geometry	EM field
physics/geometry	**single**	**single-element**	**simple**	**none**
physics/geometry	**multiple**	**single-element**	**simple**	**none**
material	single	composite	simple	none
mag. field	multiple	composite	simple	constant
mag. field	multiple	composite	simple	variable
full	multiple	composite	complex	variable

github.com/celeritas-project/benchmarks

Challenge: define problems to meet above goals ***and*** *apply to real life*

Performance measurements

- Problem definitions
  - Not all codes will be able to run same complexity/feature set
  - Scoring/diagnostics and I/O will affect runtime
  - Shared repository with inputs and tested configurations
  - Validation tie-in for accuracy (answers must agree “well enough”)
- Hardware choices and cross-architecture comparison
  - Commodity hardware or HPC (Depends on targeted use case?)
  - Portability across GPU/accelerator architectures
  - Cost of purchase vs power consumption

Challenge: consensus and community buy-in

Inter-code integration

- Input parameters and problem definitions
  - Geometry processing: GDML vs in-memory construction
  - Physics settings and material translations
  - External cross section data
  - Preprocessing cross sections

Opportunity: new HSF-led codes

- Output and workflow integration
  - Output performance can be a bottleneck for fine-grained scoring
  - Substantial potential gains for avoiding CPU interfaces for GPU workflows
  - ML training using GPU-generated hits

Challenge: HPC I/O

Challenge: performant workflow integration

End goals

- How much performance gain is needed to justify:
  - Making workflow builds more complicated (adding new codes)?
  - Purchasing new hardware (GPUs)?
- A “fresh start”
  - Justification to refactor physics for greater performance and maintainability
  - Revisit 30+ years of implementation decisions (perhaps based on older hardware, compilers)

Challenge: meeting goals and convincing sponsors

Further development in Celeritas

- Missing features
  - Element selection for discrete interactions
  - Multiple propagator/msc along-step kernels (neutral vs charged)
  - Reproducible track IDs for “MC truth” output
- Physics validation
  - Internal validation test suite
  - Community-driven test problems
- Performance improvements
  - Partition rather than mask threads
  - Search for hot spots (`rotate`: heavy register+memory+ops)
  - Single-precision in select kernels for improved bandwidth, flop rate, occupancy
  - Tabularize “on the fly” data to reduce code paths

Challenge: prioritization

Acknowledgments

- ECP

  This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

- OLCF

  This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

- Ben Morgan and the AdePT/HSF/Geant4 teams