Towards GPU Accelerated Simulation of Optical Calorimetry

Celeritas Core Team:

Elliott Biondo [1] Julien Esseiva [2] **Hayden Hollenbeck** [3] Seth R Johnson [1]
Soon Yung Jun [4] Guilherme Lima [4] Amanda Lund [5] Stefano Tognini [1]

Celeritas Core Advisors:

Tom Evans [1] Philippe Canal [4] Marcel Demarteau [1] Paul Romano [5]

[1]Oak Ridge National Laboratory [2]Lawrence Berkeley National Laboratory [3]University of Virginia

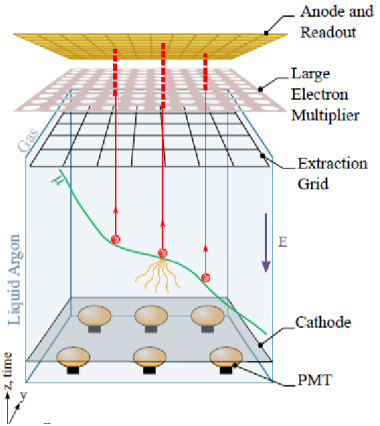
[4]Fermi National Accelerator Laboratory [5]Argonne National Laboratory [6]University of Warwick

Coordination Panel for Advanced Detectors November 20th, 2024

Motivation

Detectors measuring **scintillation** and **Cerenkov** light are abundant!

- Liquid Argon TPCs (DUNE, SBND)
- Liquid Xenon TPCs (LZ)
- Gadolinium-Doped water Cerenkov detectors (ANNIE, Super-K)
- Dual readout calorimetry (CalVision)



LArTPC detector design principle [arxiv:2103.06395]

Problem: Optical Physics in Geant4

Geant4:

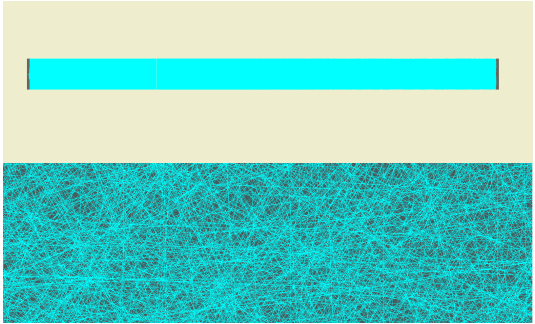
- Multithreaded on CPU
- Tracks evolve in independent threads
- Throughput scales with number of cores

HPC cluster available CPU cores: ~ 64

- $\sim 10,000$'s Cerenkov photons per event
- $\sim 100,000$'s scintillation photons per event

Doesn't scale well enough to run full optical simulations.

Example: 1 GeV muon incident on PWO_4



Zoomed: Geant4 optical photon tracks

- $\sim 20,000$ Cerenkov photons
- $\sim 100,000$ scintillation photons

Optical Physics Speed-Up Efforts

Fast Parameterizaitons

- Precompute detector response for single photons
- Scintillation well approximated (isotropic, constant spectrum)
- Cerenkov poorly approximated (highly velocity and direction dependent)
- Still need to simulate the initial parameterization

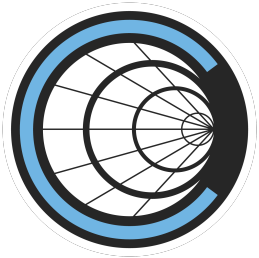
GPU Offloading

Pass initial photons to GPU simulations instead of a parameterization.

- Opticks: uses NVIDIA OptiX Engine
- Mitsuba3: just-in-time physics based renderer

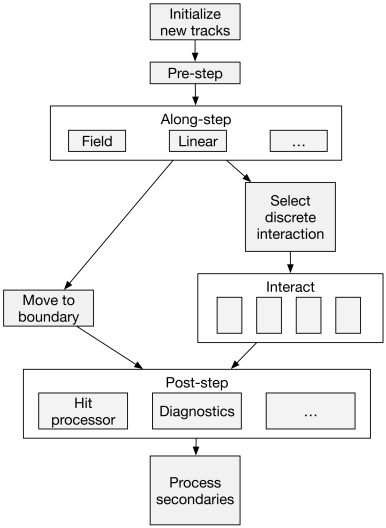
Celeritas: GPU Accelerated Detector Simulation

- **GPU accelerated**, CPU reproducible
- **Full fidelity** Monte Carlo detector simulation
- **Automated Geant4 integration** (geometry, physics, SD)
- **Open source** and actively seeking collaborators



Celeritas: Optical Physics

- Platform portable (CPU, NVIDIA, AMD, Intel)
- Separate main and optical tracking loop
  - Fine-tune optimizations for high count optical tracks
- Various integration methods
  - Standalone Celeritas full simulation
  - GPU offloading for Geant4 optical photons
  - Fast parameterization generation
  - Replace Geant4 optical physics



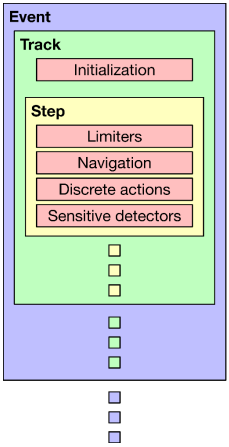
GPU Acceleration with “Actions”

Track independence allows loop reordering

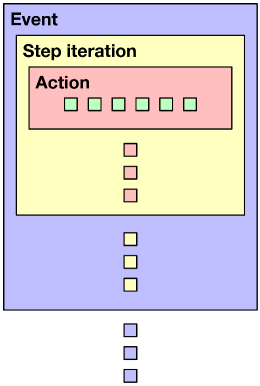
- Geant4: Event → Track → Steps
- Celeritas: Event → Steps → Tracks

Generalize steps to “actions” framework

- Loops over tracks on CPU or GPU
- Supports dynamic per-track state
- Customization point for GPU acceleration



CPU stepping loop

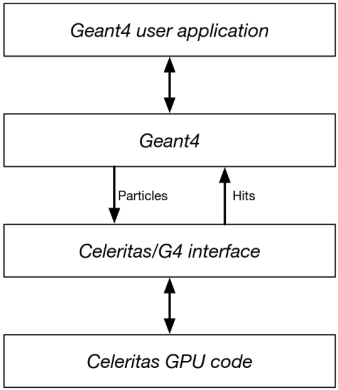


GPU stepping loop

Geant4 Integration

Goal: Drop Celeritas into existing Geant4 workflows with minimal users changes

- Directly import G4 data
- Geometry translation to VecGeom / ORANGE (Oak Ridge Advanced Nested Geometry Engine)
- Tracks offloaded to Celeritas
  - Hooks into G4's user managers
  - Selective offloading for mixed Celeritas / G4 runs
- Sensitive detector "hits" sent back to G4



Optical Physics Capabilities

Generators (implemented):

- Cerenkov and scintillation
- Runs in the main physics loop
- Create a small distribution struct
  - Pass to the optical loop
  - Generate photons from the distribution

Behavior looks like usual offloading!

- G4 and Celeritas distributions handled the same

Currently implemented:

- Absorption
- Rayleigh scattering

Planned:

- Wavelength shifting
- Mie scattering
- Surface optical physics

Proof-of-Concept: Photon Generation

- Single 10 MeV electron primary in a LAr sphere
  - Roughly **300 million** optical photons total
  - AMD EPYC 7532 (64 cores) vs Nvidia A100
- Only **generating** optical (scintillation and Cerenkov) photons, **not** tracking
  - `celer-sim` (standalone Celeritas): kill the optical photons immediately
  - `celer-g4` (Geant4 front-end): immediately kill optical photons via user tracking action

	CPU threads	Runtime (s)
Geant4	1	204.5
Celeritas CPU	1	99.3
Celeritas CPU (OMP)	64	5.7
Celeritas GPU	1	1.3

OMP: OpenMP Track level parallelism

150 CPU : 1 GPU performance tracks with previous results

Conclusion

- Celeritas is working towards full optical simulations
  - GPUs critical for performance
  - Platform portability
  - Geant4 compatability
- Still a work in progress
- Volumetric optical physics by end of year
- Surface physics by mid-2025
- Concurrent development of muon physics, neutrino physics, geometry, ...
- Actively coordinating with experiments to meet their simulation requirements and goals!

Acknowledgements

Celeritas v0.5 code contributors:

- Elliott Biondo (@elliottbiondo)
- Philippe Canal (@pcanal)
- Julien Esseiva (@esseivaju)
- Tom Evans (@tmdelellis)
- Pete Heywood (@ptheywood)
- Hayden Hollenbeck (@hhollenb)
- Seth R Johnson (@sethrj)
- Soon Yung Jun (@whokion)
- Guilherme Lima (@mrguilima)
- Amanda Lund (@amandalund)
- Ben Morgan (@drbenmorgan)
- Stefano C Tognini (@stognini)

Past code contributors:

- Doaa Deeb (@DoaaDeeb)
- Vincent R Pascuzzi (@vrpascuzzi)
- Paul Romano (@paulromano)

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.

SciDAC: This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, Scientific Discovery through Advanced Computing (SciDAC) program.

NERSC: This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231 using NERSC award HEP-ERCAP-0023868.

- Code: github.com/celeritas-project/celeritas
- Documentation: celeritas-project.github.io/celeritas/dev/index.html
- Publication: doi.org/10.1051/epjconf/202429511005