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

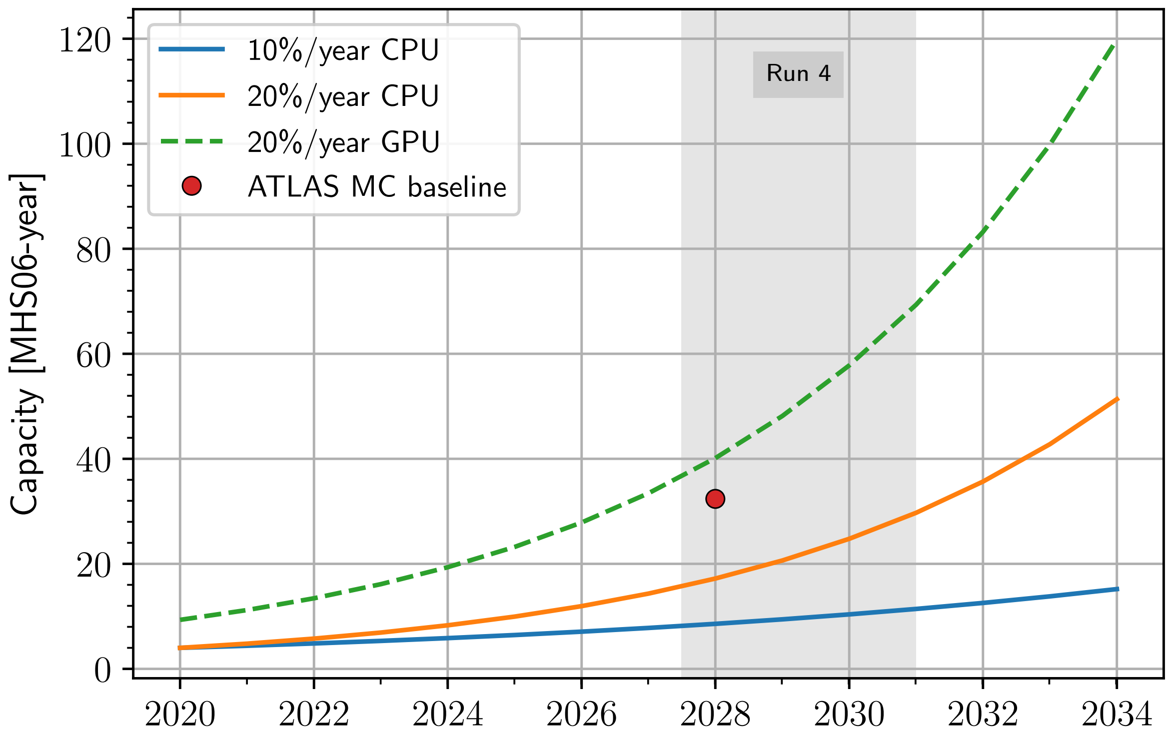
Background

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

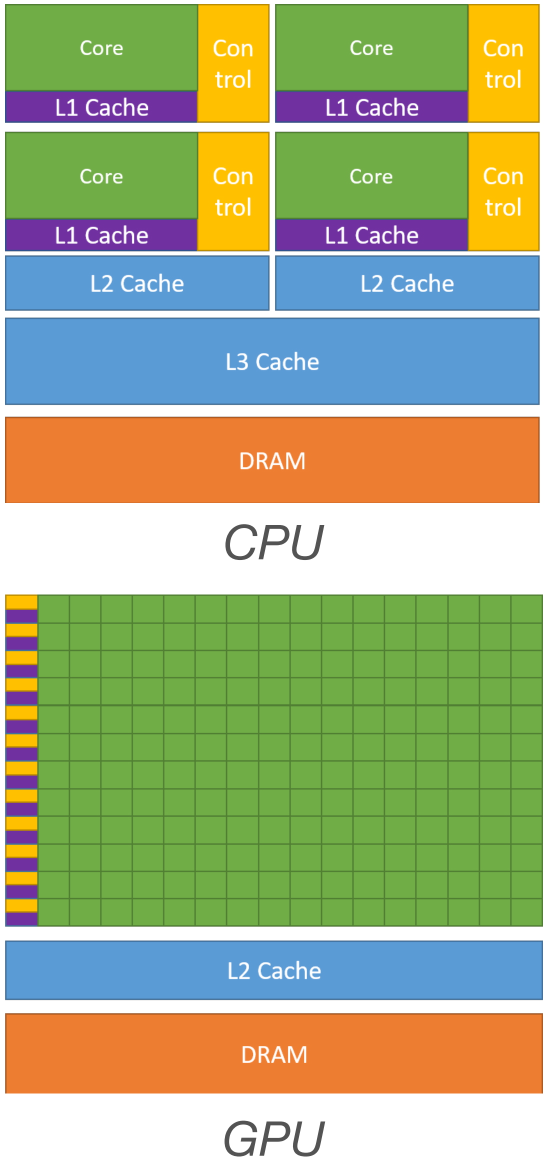
- High Luminosity upgrade means 10× higher sampling rate
- More detector data means more simulations needed
- Tens of millions of “equivalent 2006-era CPU hours” for analysis
- 20–25% is from full fidelity MC



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

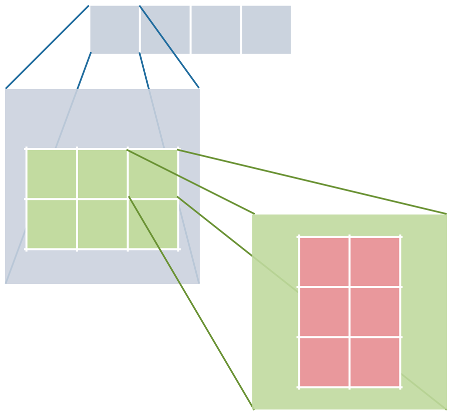


https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

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

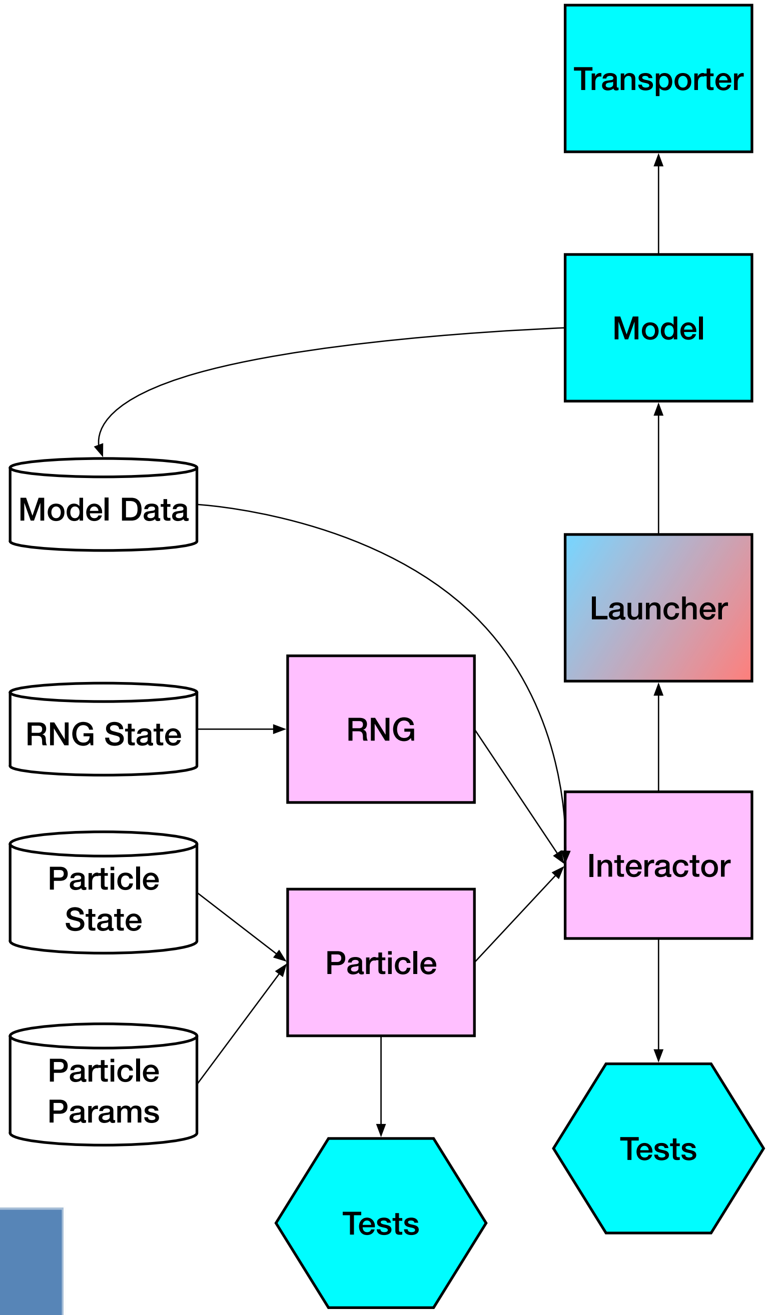
Structured grid data



Monte Carlo data

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

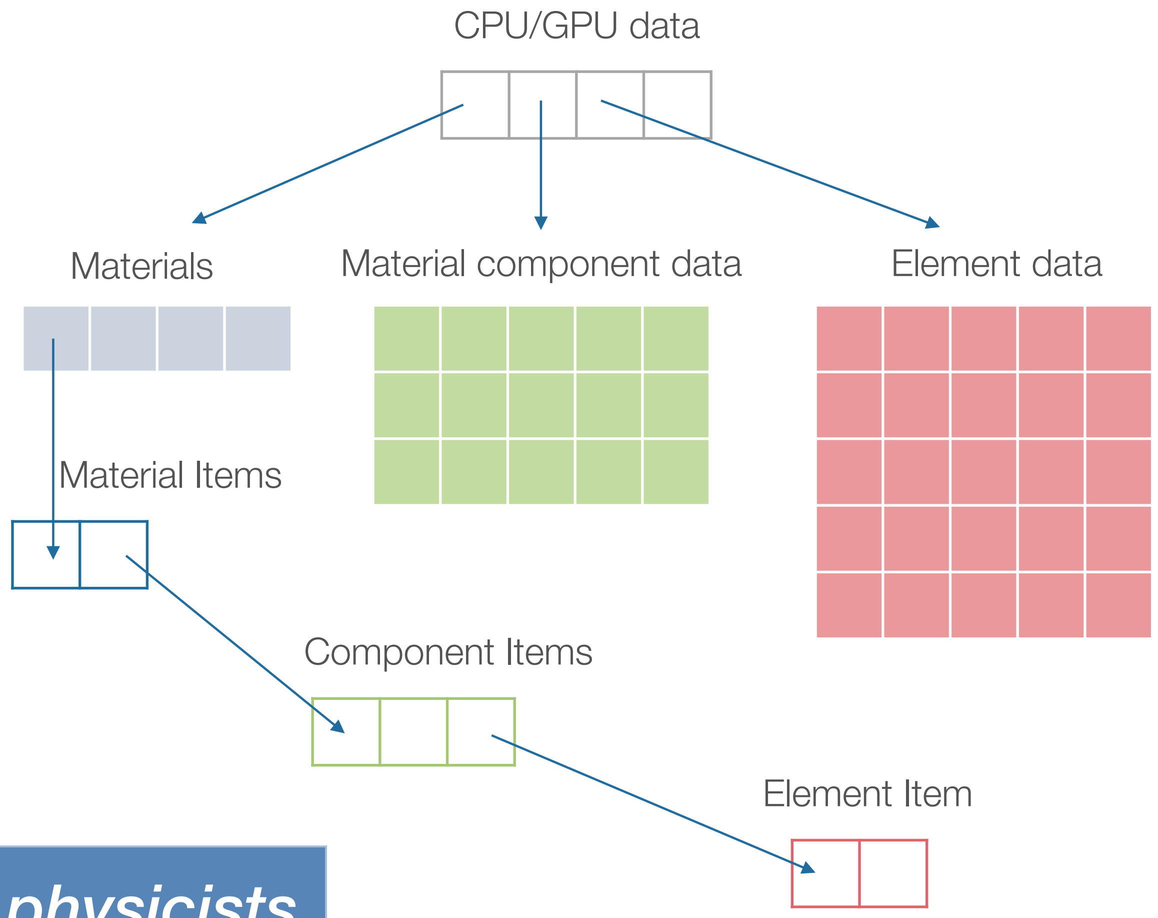


Easily refactored for new architectures, data models, performance

Features

Memory model for hierarchical data

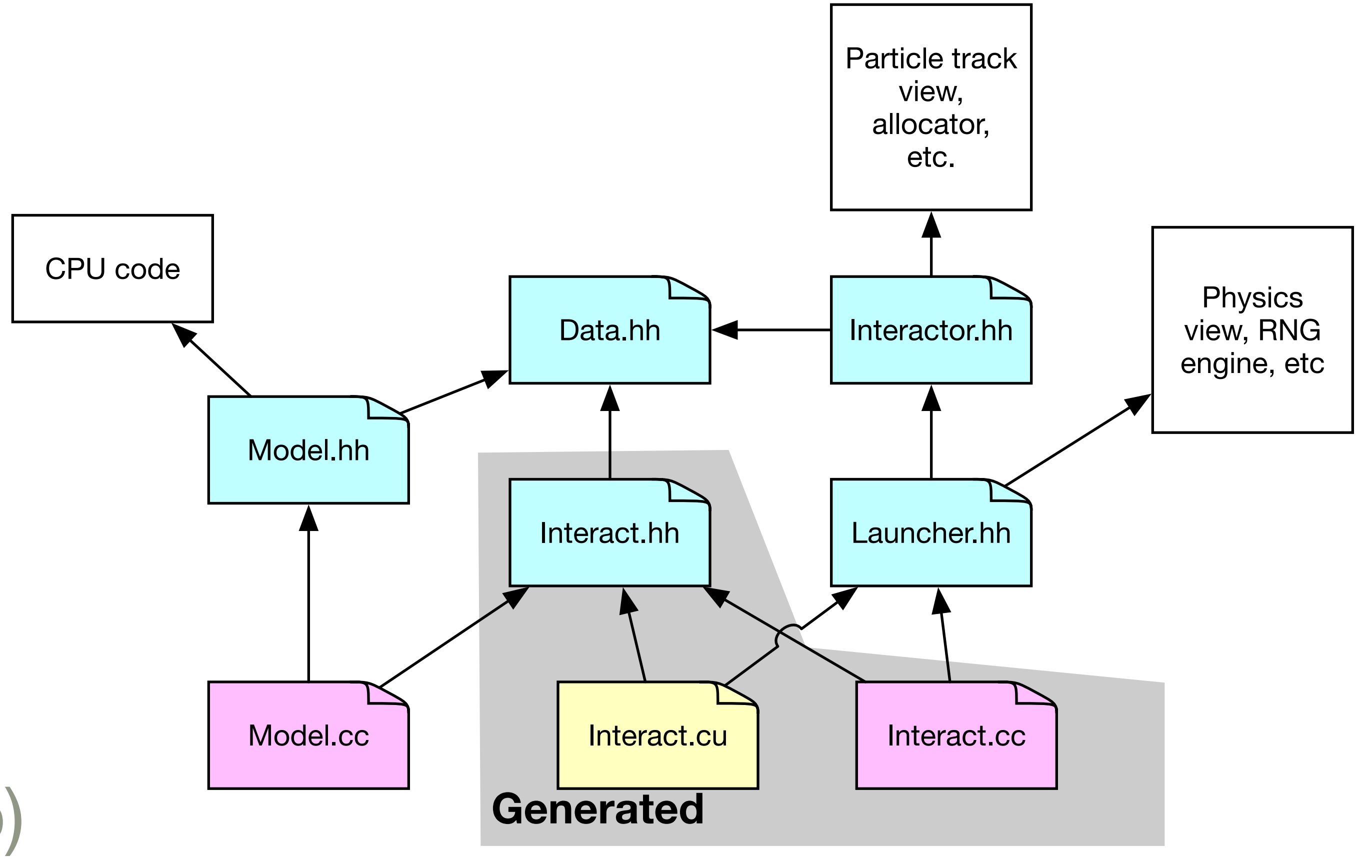
- Define data structures *once*
- Easily assemble data on CPU
- Data and execution on both CPU and GPU
- Single-line data transfer



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

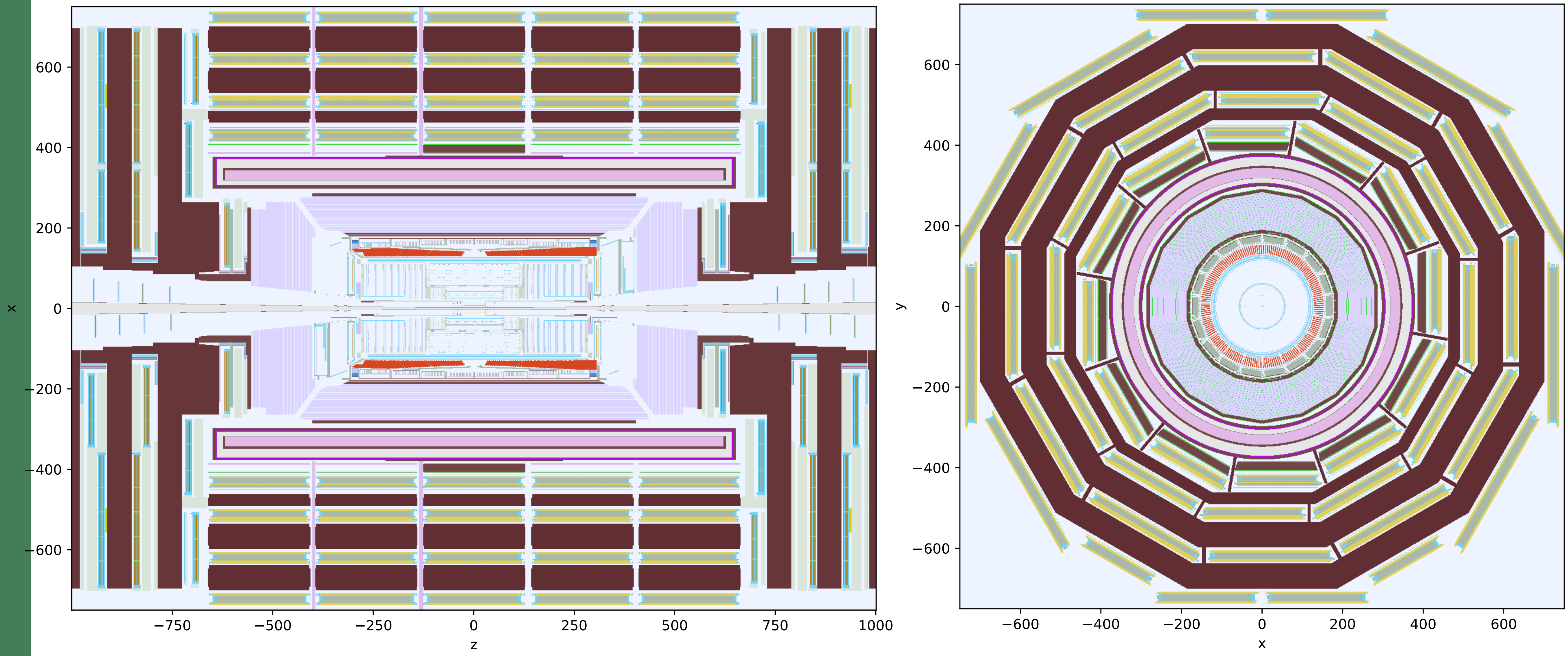
Multi-architecture portability

- 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

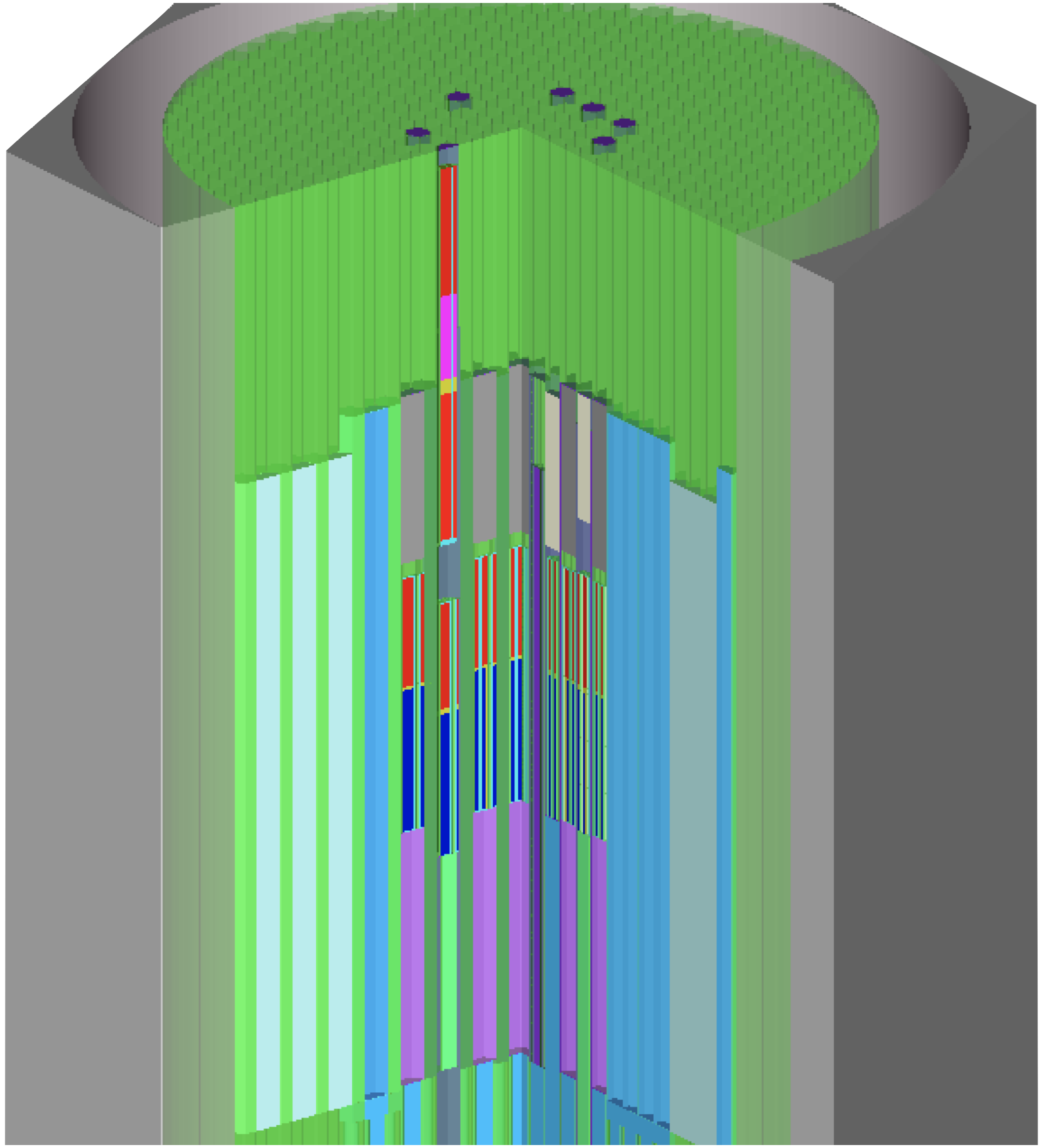
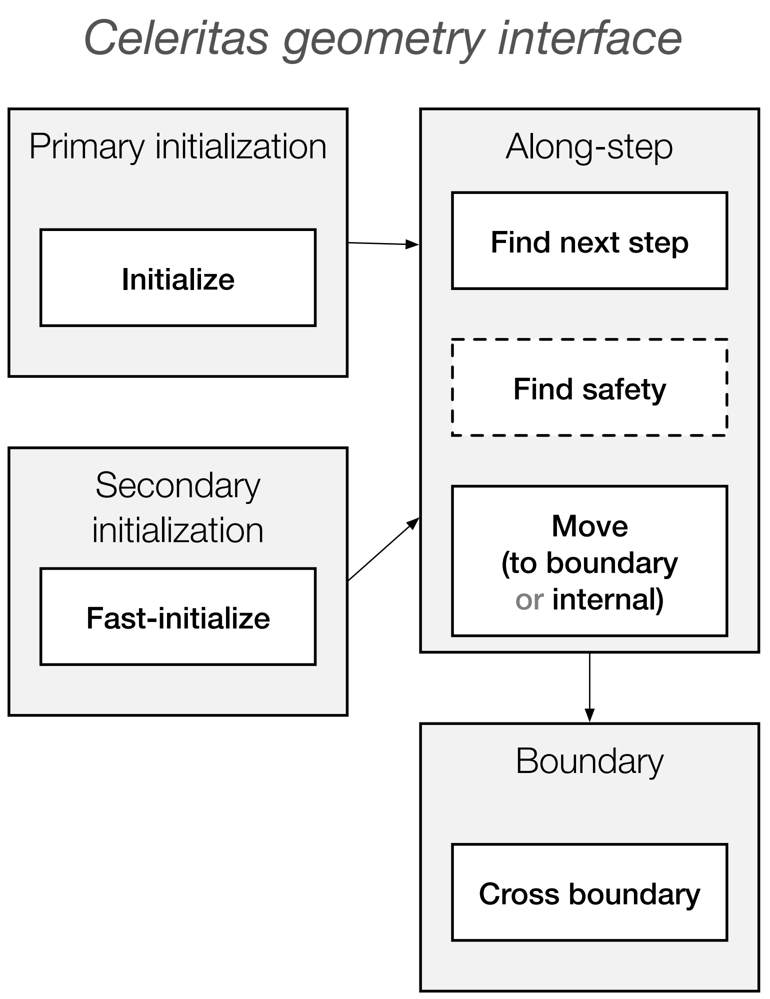
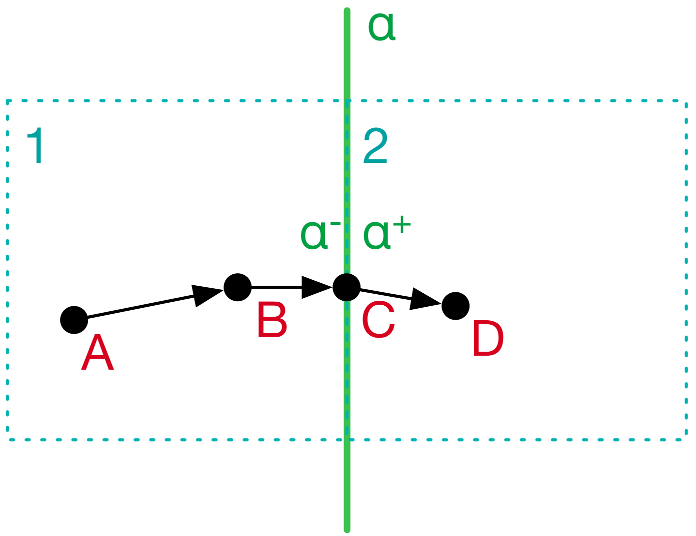


Image credit: Steve Skutnik (ORNL)

ORANGE surface-based tracking methodology

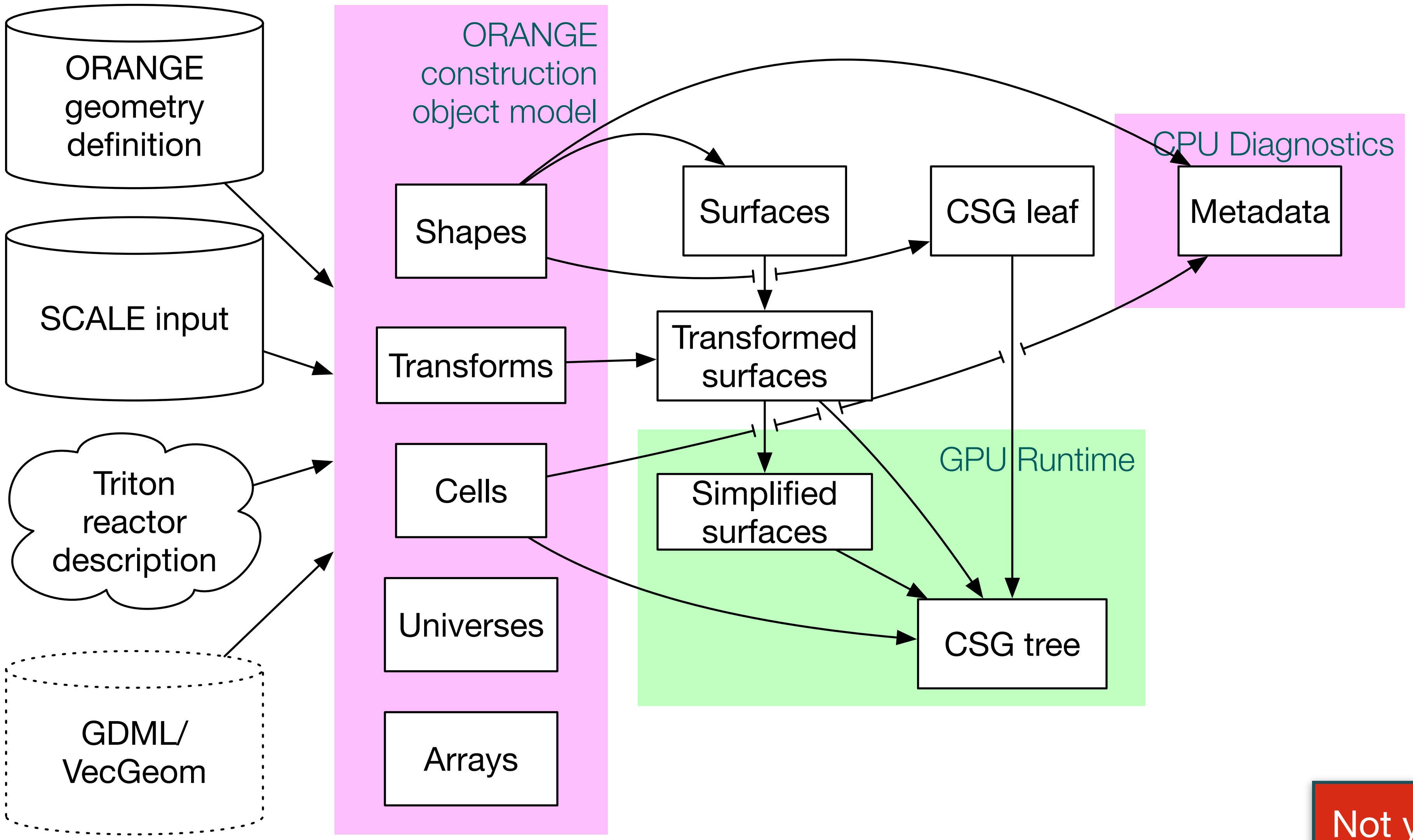


	Position	Volume	Surface+Sense
Initialize	A	1	—
Find step	A	1	—
Move internal	B	1	—
Move to bdy	C	1	α inside
Cross bdy	C	2	α outside
Move internal	D	2	—



Oak Ridge National Laboratory

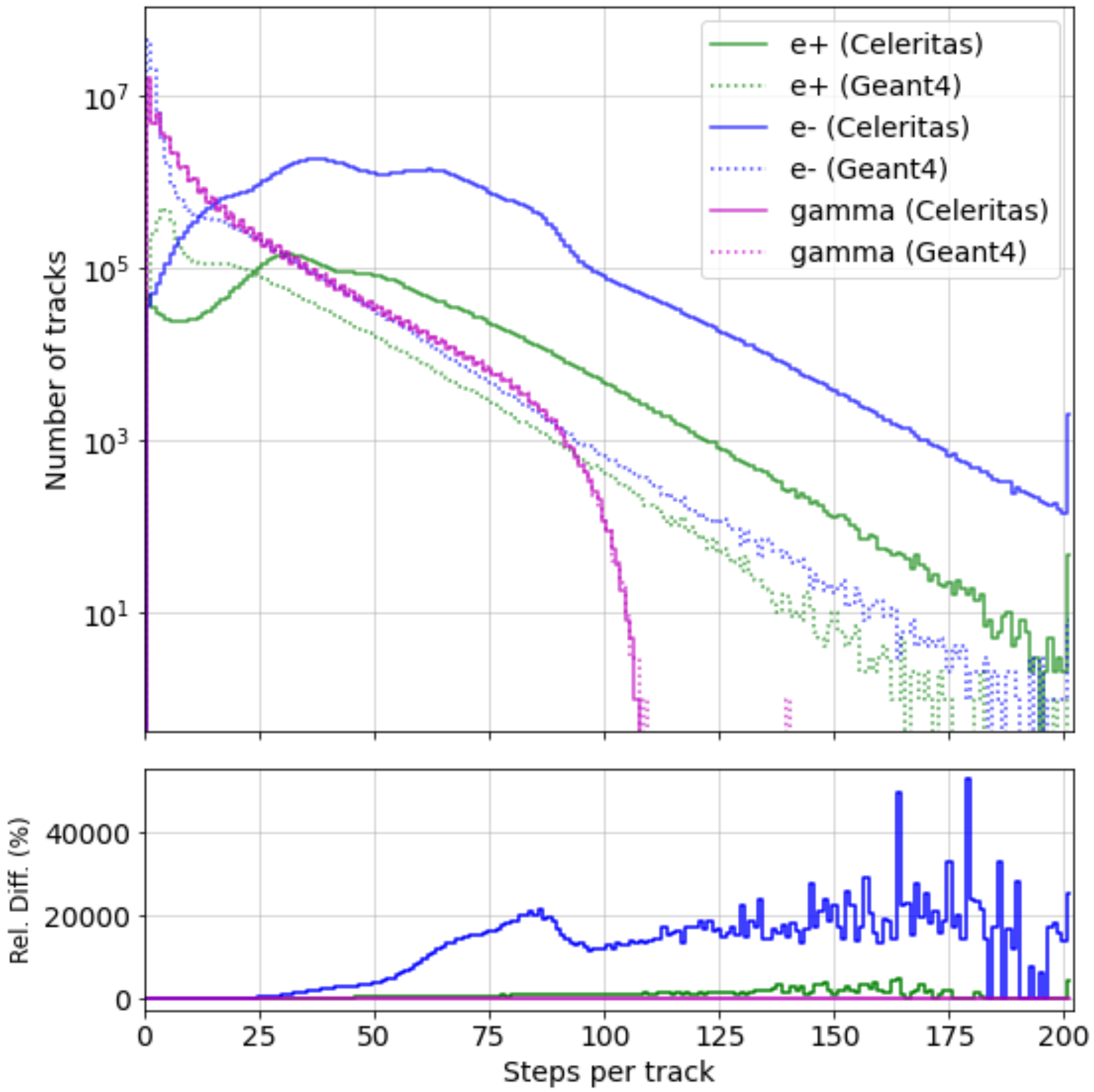
ORANGE surface/volume construction



Not yet available in Celeritas ORANGE

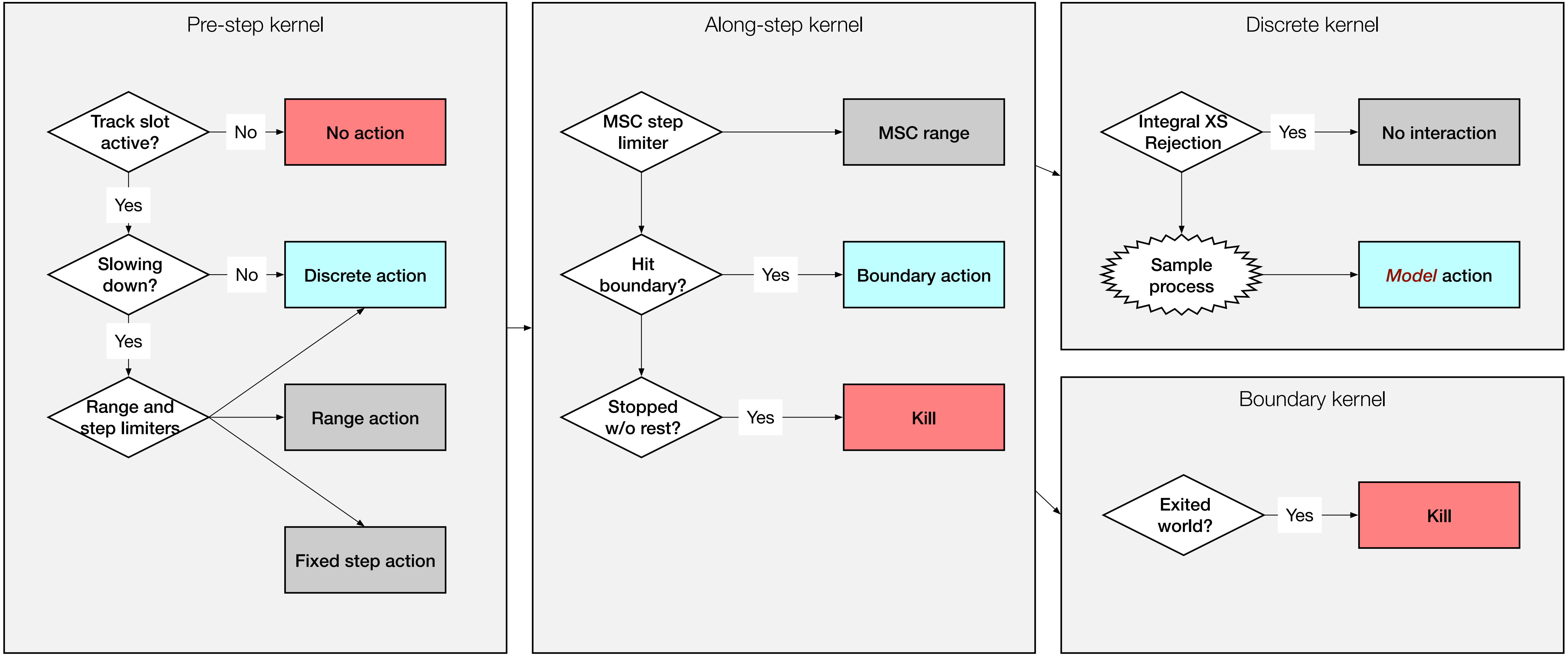
Standard EM physics

Particle	Process	Model(s)
γ	photon conversion	Bethe–Heitler
	Compton scattering	Klein–Nishina
	photoelectric effect	Livermore
	Rayleigh scattering	Livermore
$e^{\pm}$	ionization	Møller–Bhabha
	bremsstrahlung	Seltzer–Berger, relativistic
	pair annihilation	EPlusGG
	multiple scattering	Urban
$\mu^{\pm}$	muon bremsstrahlung	Muon Bremsstrahlung



Urban MSC comparison (in progress!)

Transport loop and control flow



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 VGDML 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)

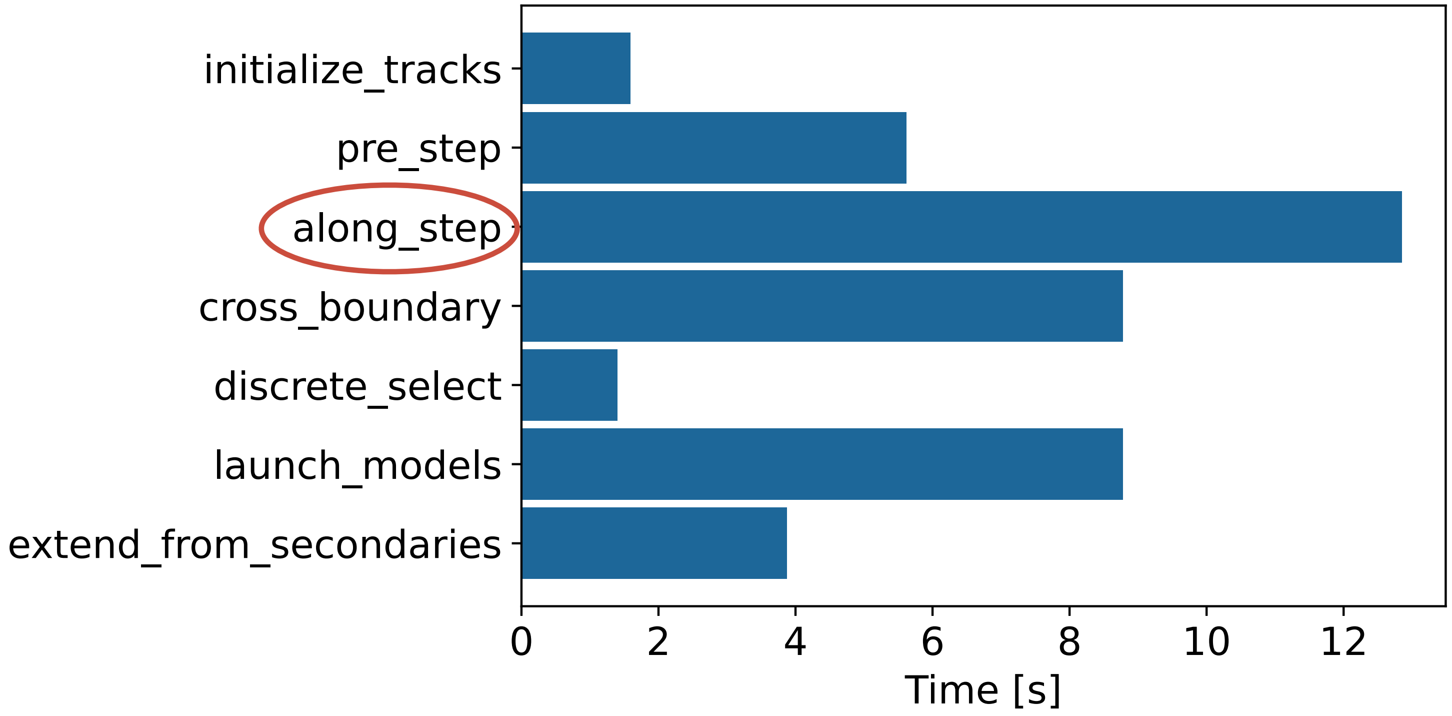
	geo	arch	mean	σ
Geant4 10.7.1	Geant4	CPU	2.9	0.1170
Celeritas 8d83ebab (29 Apr 2022)	ORANGE	CPU	2.09	0.0192
		GPU	0.046	0.0012
	VecGeom	CPU	1.95	0.0352
		GPU	0.0627	0.0004

Number of primaries per run

Geant4	Geant4	CPU	1E+04
Celeritas	ORANGE	CPU	1E+03
		GPU	1E+05
	VecGeom	CPU	1E+03
		GPU	1E+05

Detailed timing

```
extend_from_primaries                  ▷ Copy primaries to device, create track initializers
while Tracks are alive do
    initialize_tracks                  ▷ Create new tracks in empty slots
    pre_step                           ▷ Sample mean free path, calculate step limits
    along_step                         ▷ Propagation, slowing down
    boundary                           ▷ Cross a geometry boundary
    discrete_select                    ▷ Discrete model selection
    launch_models                      ▷ Launch interaction kernels for applicable models
    extend_from_secondaries            ▷ Create track initializers from secondaries
end while
```



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)