Celeritas: EM physics on GPUs

Seth R Johnson

Celeritas code lead

Celeritas core team:

Elliott Biondo *(ORNL),* Philippe Canal *(FNAL)*,
Julien Esseiva *(LBNL),* Tom Evans *(ORNL),*
Seth R Johnson *(ORNL),* Soon Yung Jun *(FNAL)*,
Guilherme Lima *(FNAL)*, Amanda Lund *(ANL)*,
Paul Romano *(ANL)*, Stefano Tognini *(ORNL)*

Celeritas SciDAC PI:

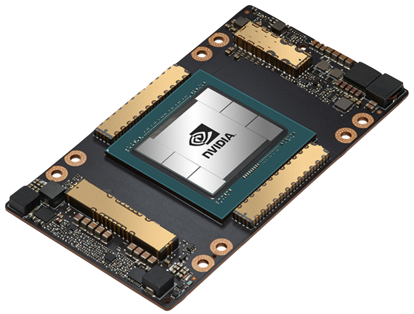
Marcel Demarteau *(ORNL)*

CalVision meeting
8 June, 2023
(derivative from CHEP 2023 presentation)

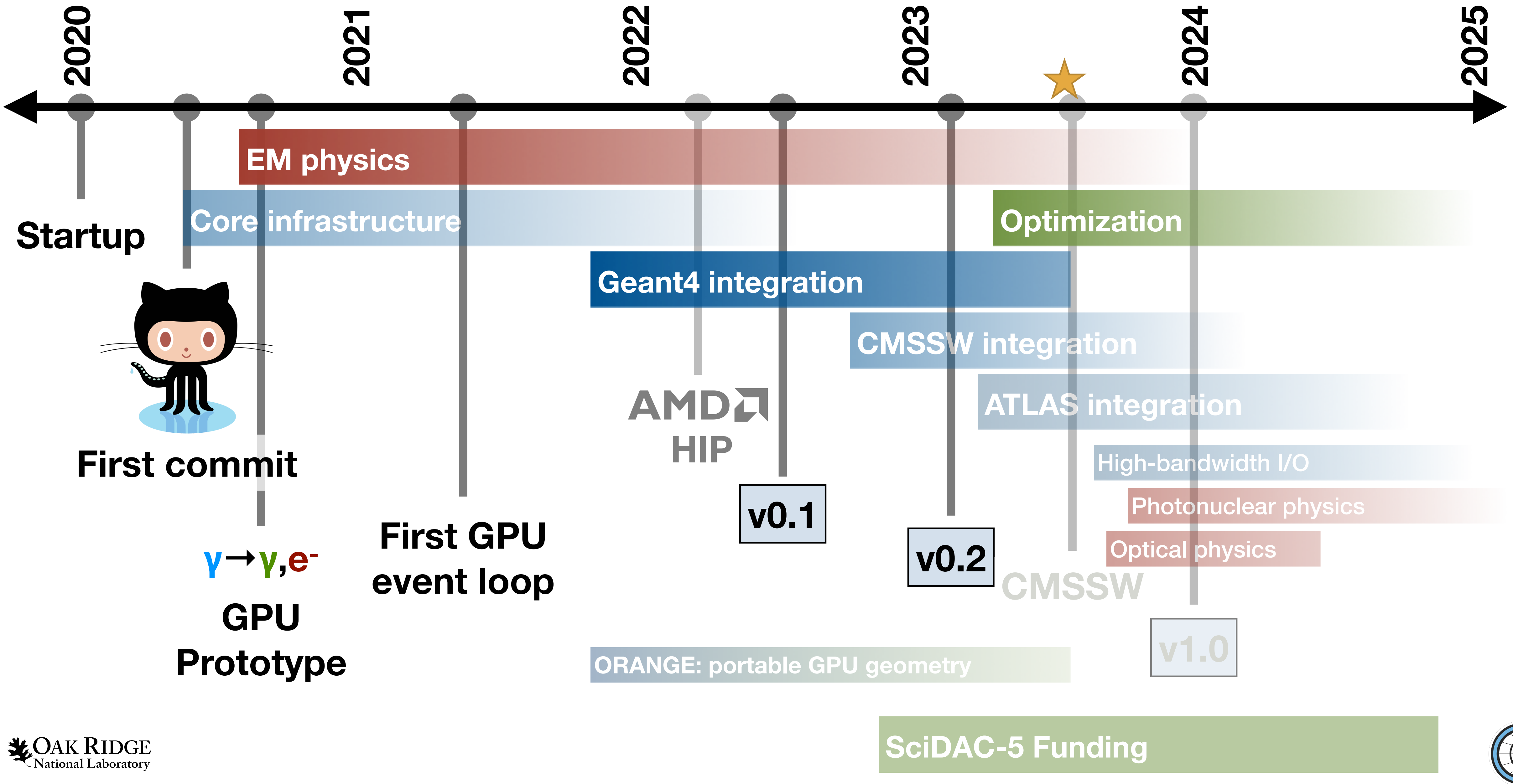
Background

Project overview

- **GPU**-focused implementation of experiment-agnostic **HEP** Monte Carlo detector simulation
- Motivated by HL-LHC computational challenges *and* by recent success in GPU MC (*ECP ExaSMR*)
- **Primary goal: accelerate production use for LHC Run 4**


Nvidia A100 GPU (Nvidia)

Present-day timeline

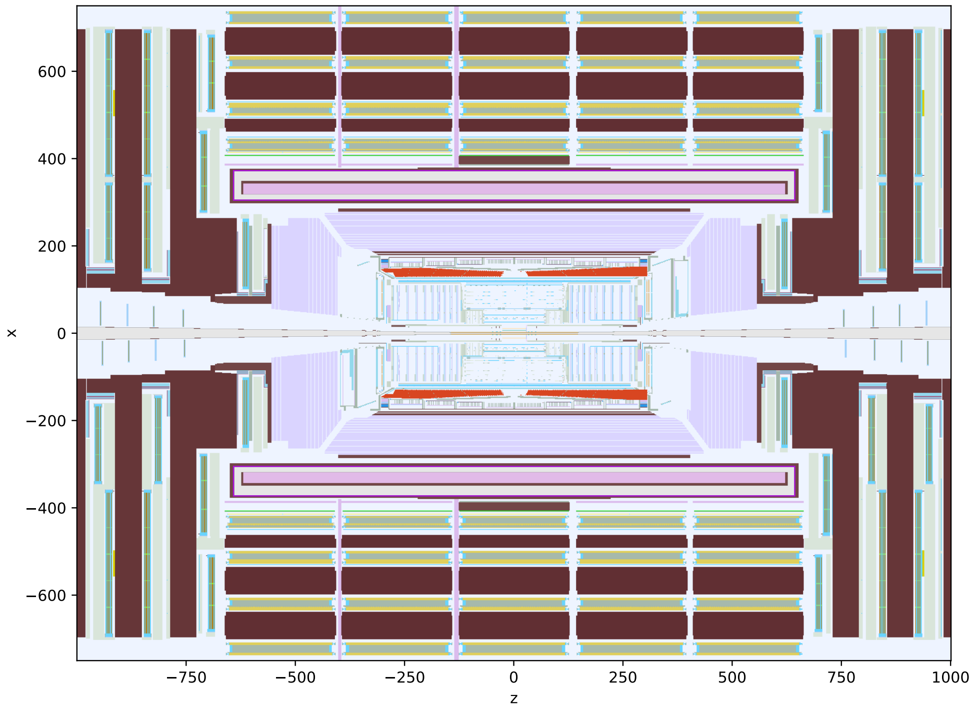


OAK RIDGE National Laboratory

Capabilities

High-level capabilities

- Equivalent to `G4EmStandardPhysics`
  …using Urban MSC for high-E MSC; only γ, e±
- Full-featured Geant4 detector geometries using VecGeom
- Runtime selectable processes, physics options, field definition
- Execution on CUDA (Nvidia), HIP* (AMD), *and CPU* devices

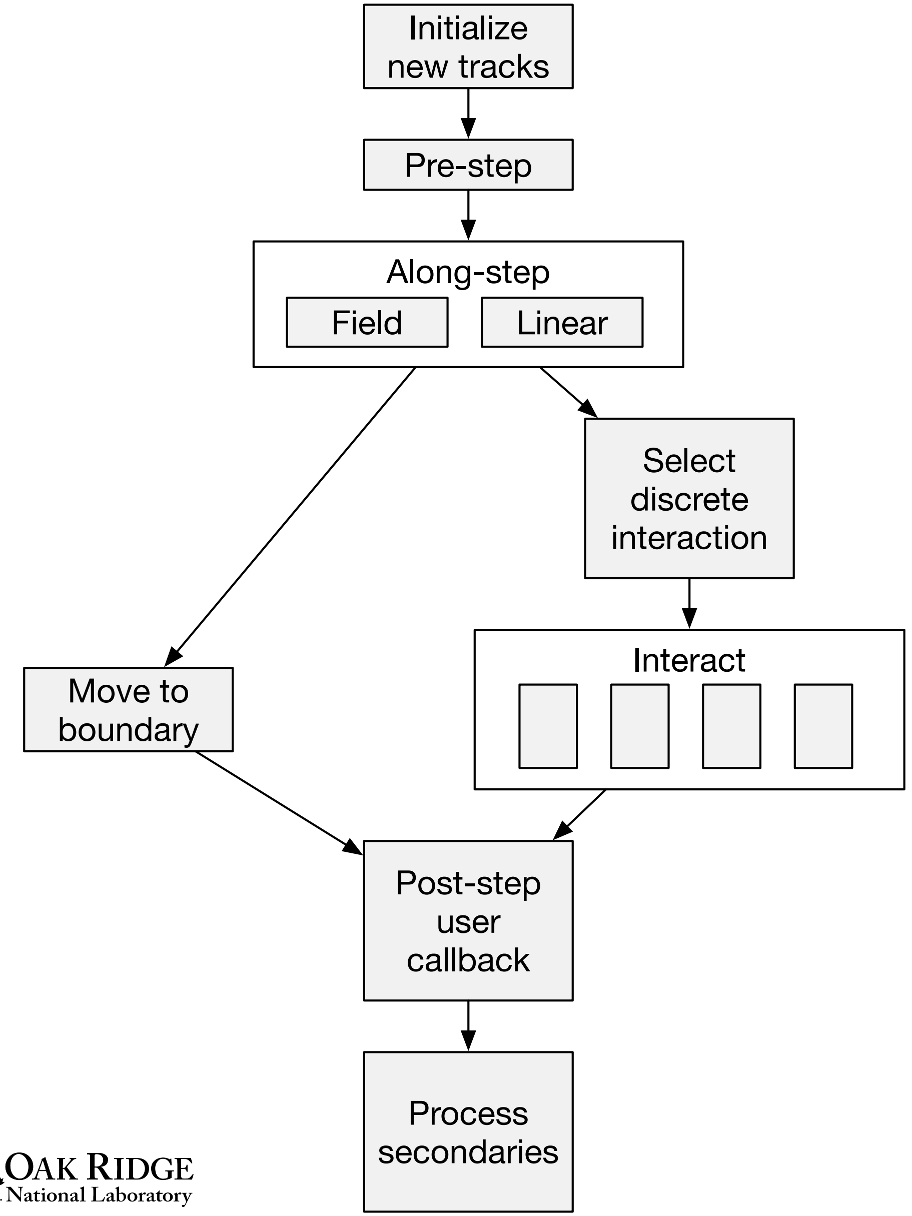


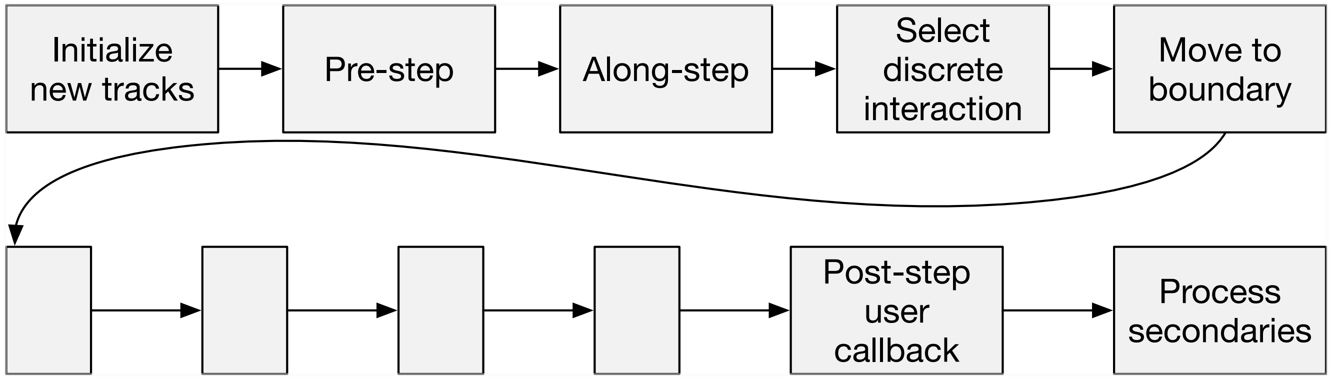
GPU-traced rasterization of CMS 2018

**VecGeom is incompatible with HIP: ORANGE GPU prototype used instead*

Verification & Validation still in progress

Stepping loop on a GPU





Topological sort: a loop over kernels

Process ~1M track batches

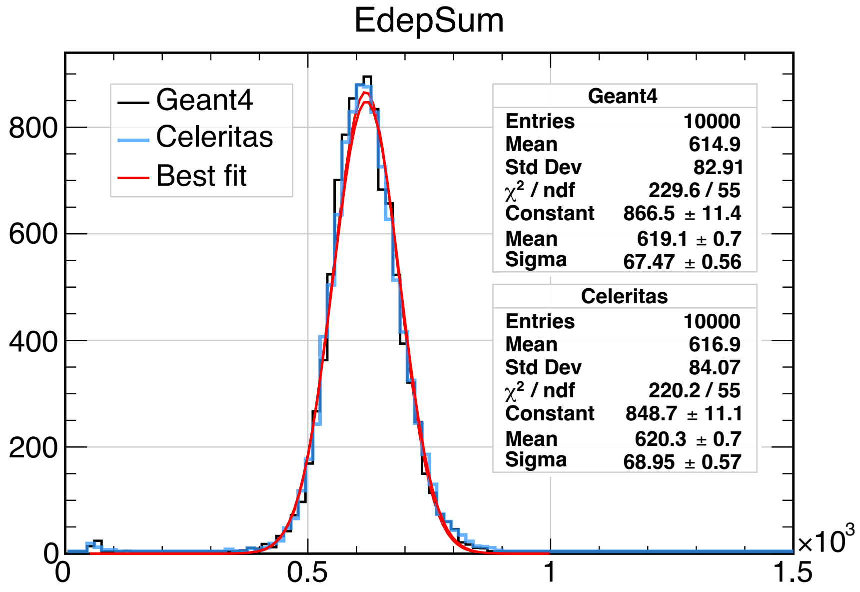
Celeritas version `0.3-dev`: Geant4 integration status

- **Imports** EM physics selection, cross sections, parameters
- **Converts** geometry to VecGeom model
- **Offloads** EM tracks from Geant4
- **Scores** hits to user “sensitive detectors”
- **Includes** GPU-optimized simple calorimeter
- **Integrates** with Geant4 10.6–11.0
- **Supports** physics/geometry/setup changes at link/run time

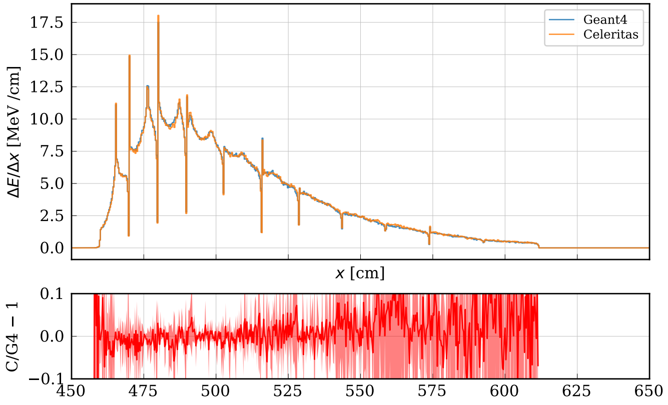
Celeritas is not designed to be a prototype code

Tilecal: ATLAS tile calorimeter test beam

- Standalone subdetector test
  - Forked from Pezzotti&Lachnit (CERN)'s work
  - 18 GeV π^+ beam, no field
  - FTFP_BERT physics
  - Primary output: energy deposition integrated over sensitive regions
- Offload e^-, e^+, γ to Celeritas
- Celeritas returns hits to user-defined `G4VSensitiveDetector`
- ~100 lines of code to integrate
- Excellent agreement



Average energy deposition with pi+ test beam



Slab-integrated energy deposition

https://github.com/celeritas-project/atlas-tilecal-integration

Framework integration status

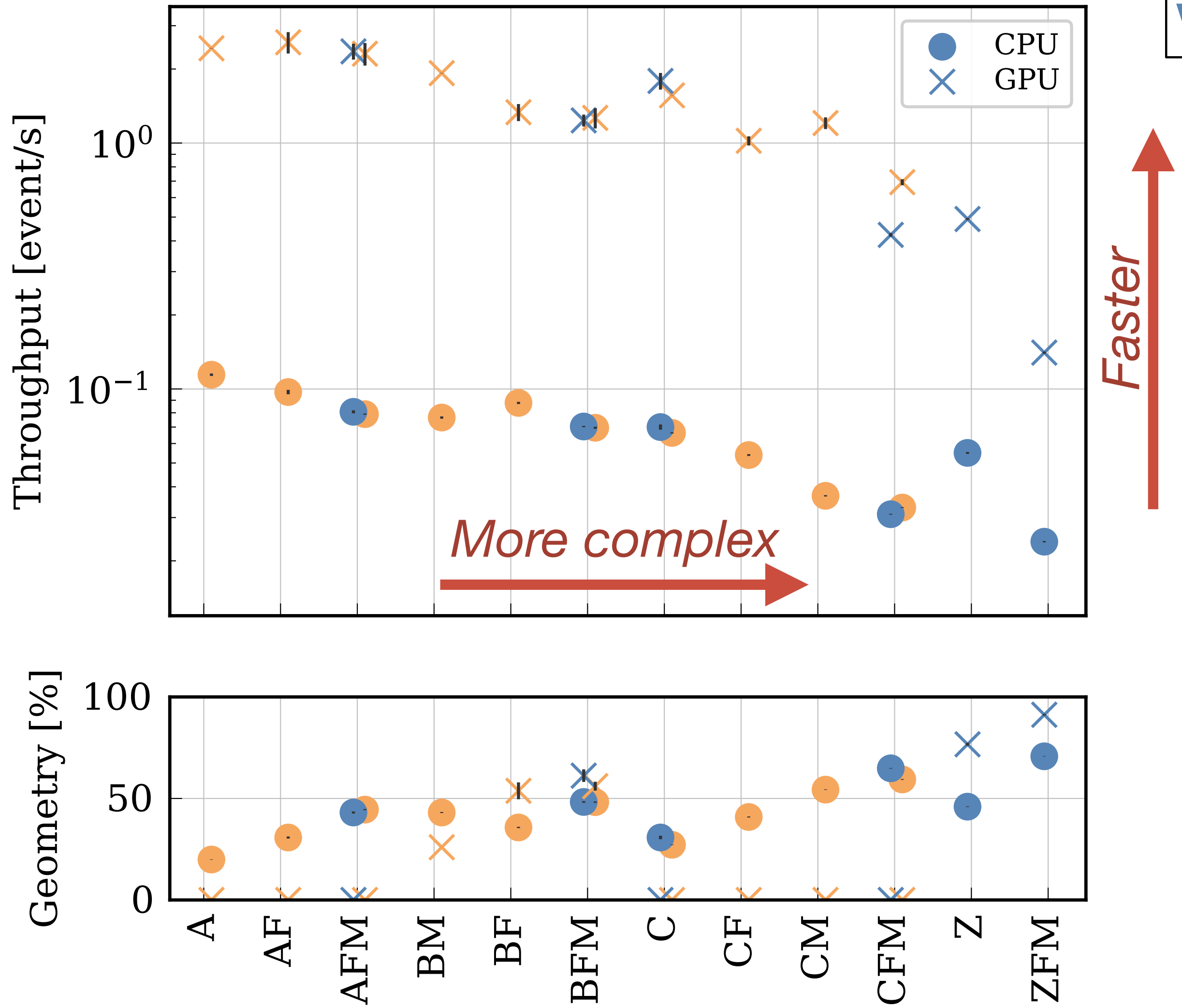
- CMS (CMSSW): offload interface implemented and running
- ATLAS (Athena): framework integration started
  - Infrastructure update for CMake compatibility: atlasexternals!1001
  - Non-custom "accordion" shape needed for VecGeom/GPU
- LHCb: seeking collaborators!
- LZ (BACCARAT): awaiting optical physics

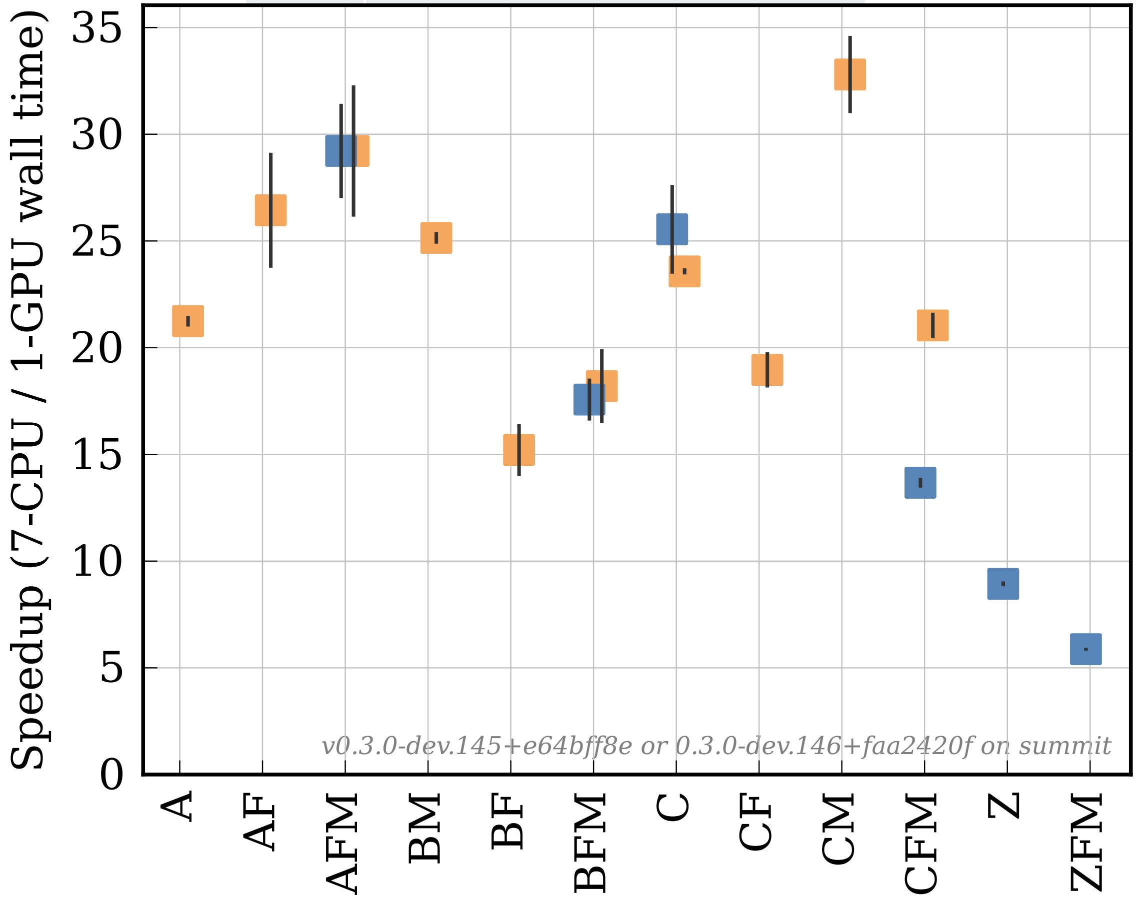
Summit GPU performance

ORANGE
VecGeom

	Problem definition
A	testem15
B	simple-cms
C	testem3
Z	cms2018

	Modifier
F	+field
M	+msc



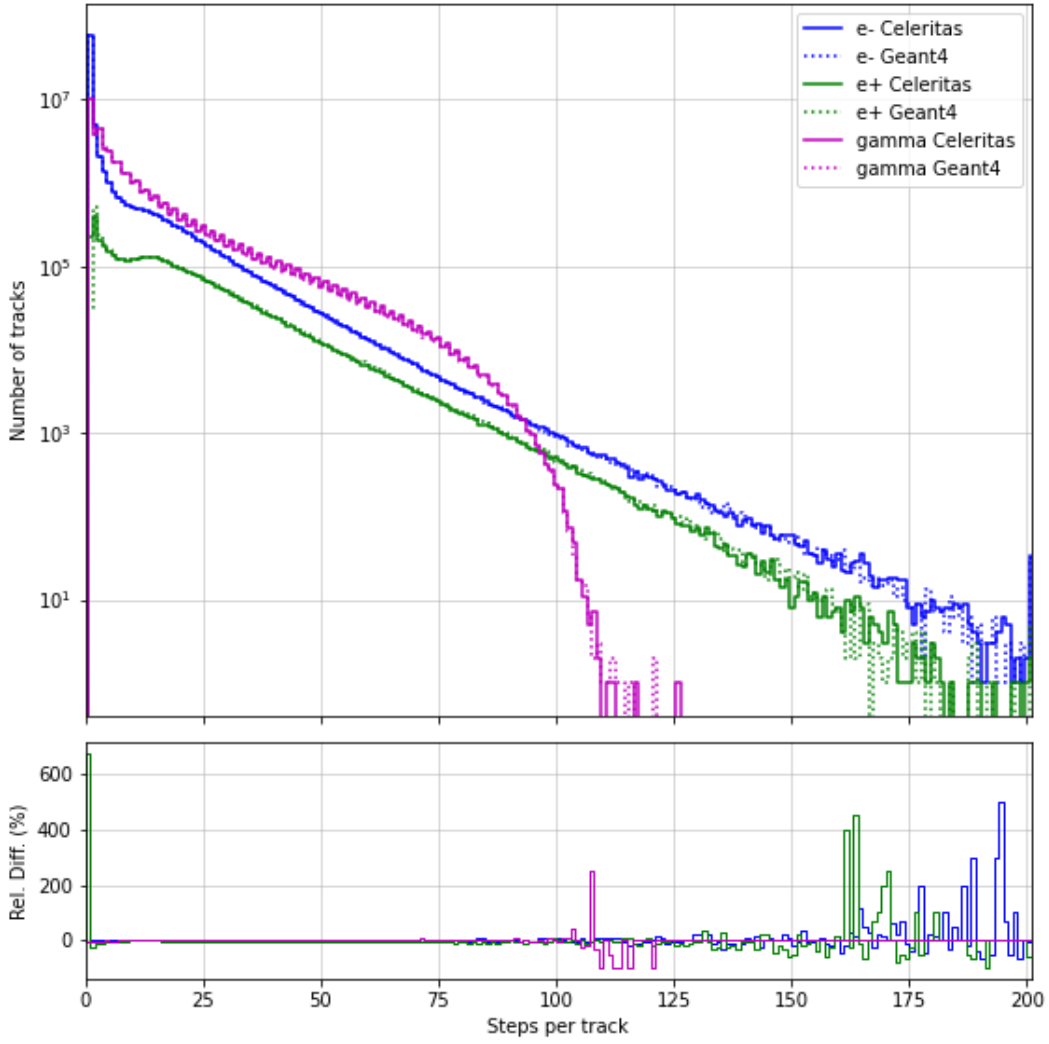


Multiply speedup by 7× for CPU:GPU equivalence

Continuing work

Validation

- Geant4 interface allows rapid comparisons
- Independent granular physics verification
- Benchmark progression problems being developed with CERN SFT (AdePT) group



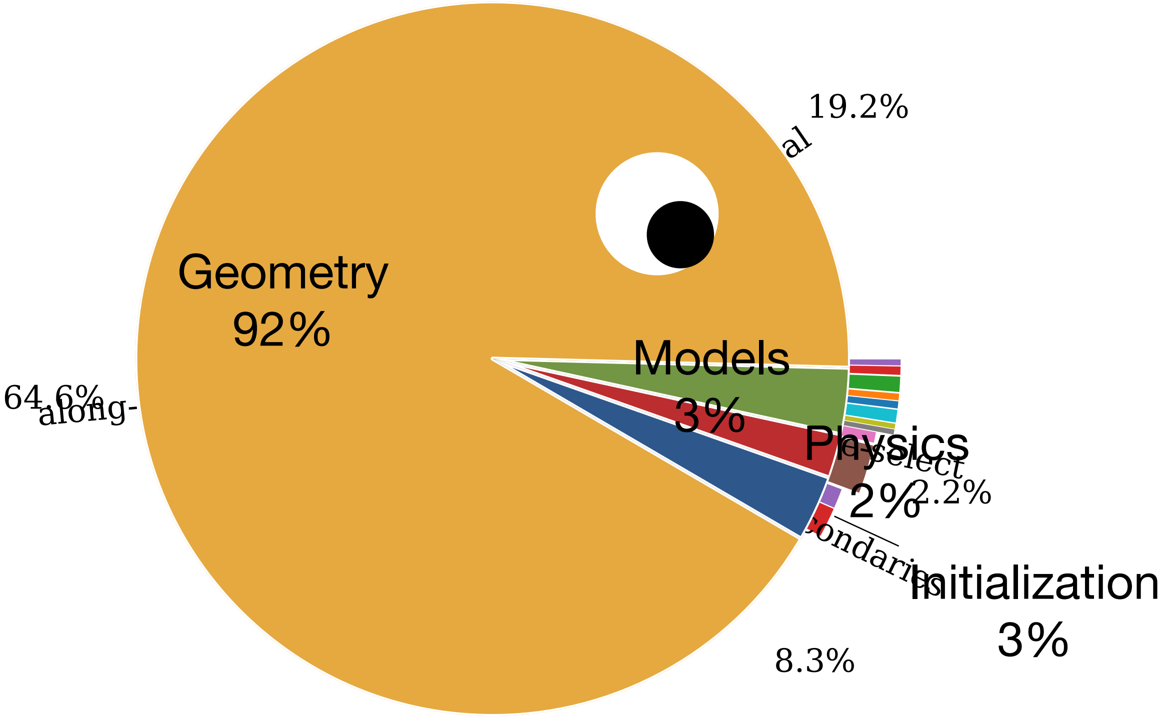
TestEM3 MSC step count verification (Amanda Lund)

Integration

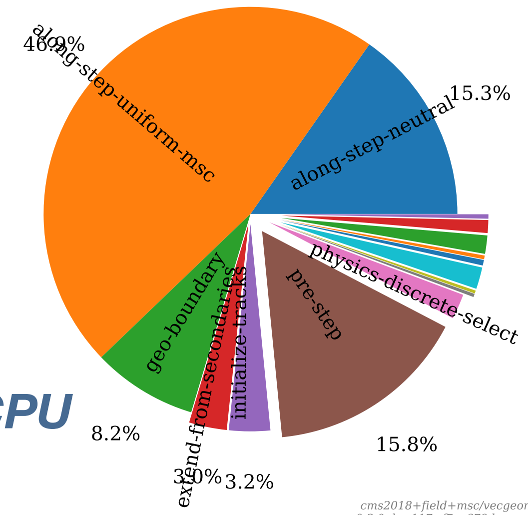
- Theoretical maximum performance gain offloading EM tracks: **~3.3×**
  (with 1000 $t\bar{t}$ events and CMS Run3 geometry)
- CMSSW offloading with RZ mapped field: **June 2023**
- ATLAS integration will require low-fidelity “accordion” for now
- Platform-agnostic optical photon acceleration in the works

Optimization

- **92%** of standalone runtime in CMS2018 is in geometry routines
- GPU native sensitive detectors
- Performance on non-HPC graphics cards still unexplored
- Goal for GPU performance for HL-LHC electron shower:
  - **2×** per watt vs CPU *(efficiency)*
  - **160×** CPU:GPU *(capacity)*



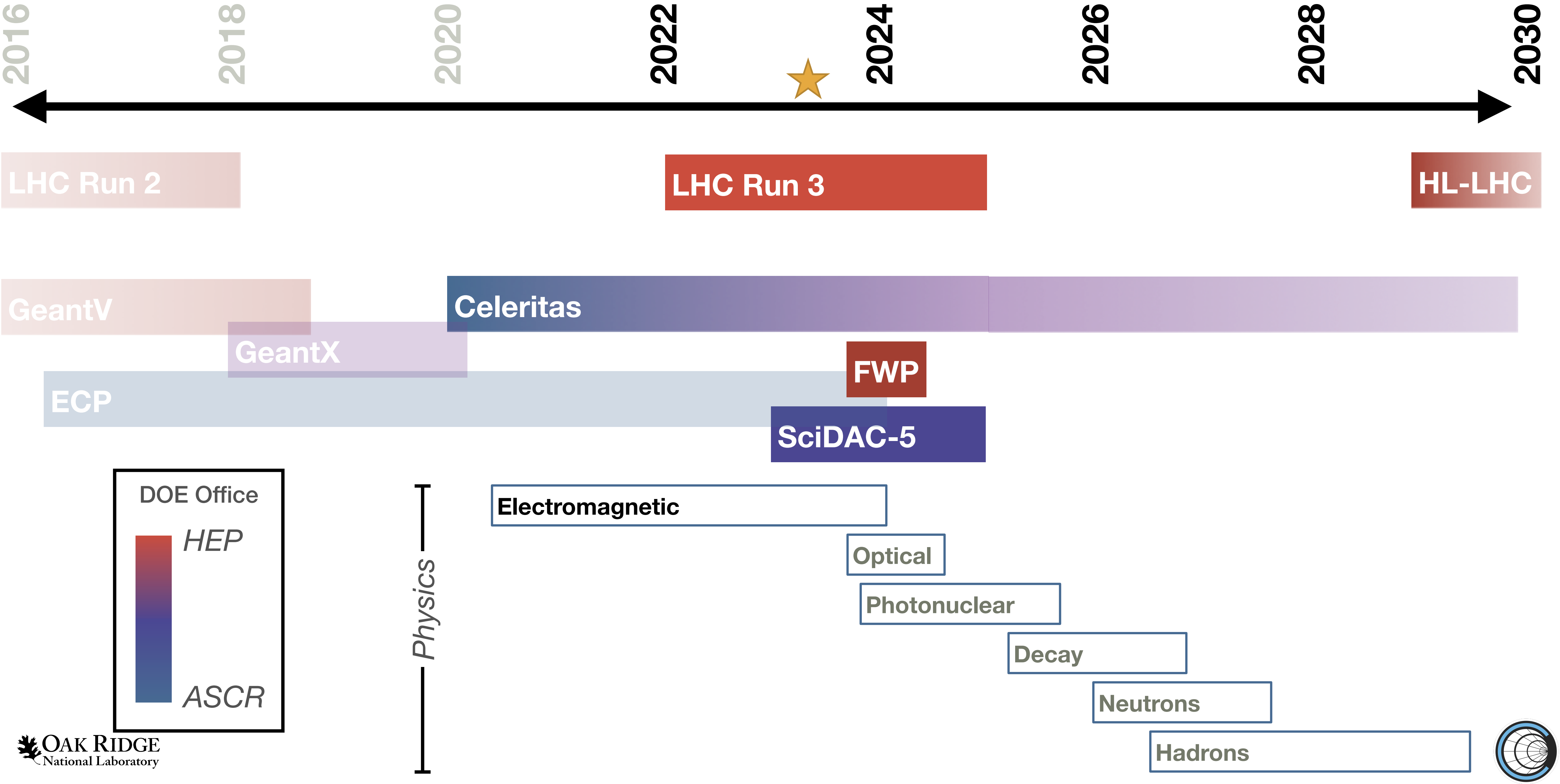
CMS2018 GPU



CMS2018 CPU

cms2018+field+msc/vecgeom/cpu v0.3.0-dev.117+f5cc679d on summit

Celeritas future timeline



Summary

- Many steps toward polished GPU detector simulation framework
- Current test problems show **~6–30× performance boost** for Celeritas detector simulation using GPUs on Summit
  (42–210× GPU/CPU core equivalence)
- Upcoming capabilities will extend our problem domain beyond LHC

Acknowledgments

Celeritas v0.2 code contributors:

- Elliott Biondo (@elliottbiondo)
- Philippe Canal (@pcanal)
- Seth R Johnson (@sethrj)
- Soon Yung Jun (@whokion)
- Guilherme Lima (@mrguilima)
- Amanda Lund (@amandalund)
- Ben Morgan (@drbenmorgan)
- Paul Romano (@paulromano)
- Stefano C Tognini (@stognini)

Past code contributors:

- Doaa Deeb (@DoaaDeeb)
- Tom Evans (@tmdelellis)
- Vincent R Pascuzzi (@vrpascuzzi)

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.

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.

https://github.com/celeritas-project/celeritas

Backup slides

Execution plan: EM physics and integration

1: Implement minimal feature set to offload EM particles

(importing Geant4 physics data, recreating G4Hit structure, implementing EM models)

2: Establish baseline performance with minimum, verified features

Standalone CMS2018 with magnetic field and full EM physics is our key problem

3: Optimize performance

- Standalone GPU performance
- Multitask/thread+GPU performance
- GPU-based sensitive detectors (calorimeters, etc)

Modular software design and robust testing is key

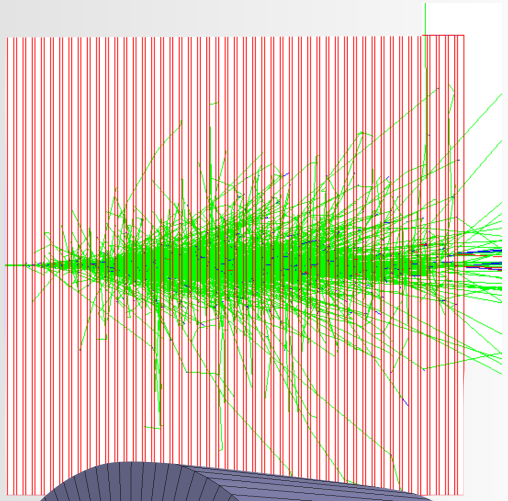
Extensibility

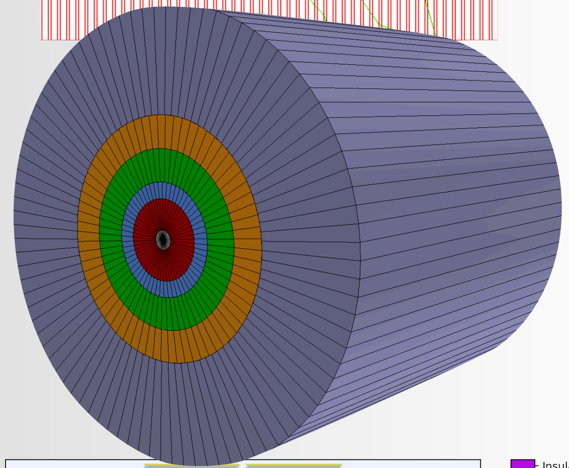
- New models and detector integrations can be added **at link time**
  (maybe even `dlopen` in the future)
- Robust integration pathways for frameworks and applications
- Code is amenable to major refactoring with minor changes
  - Modular structure
  - Composition-based classes
  - Data-oriented design

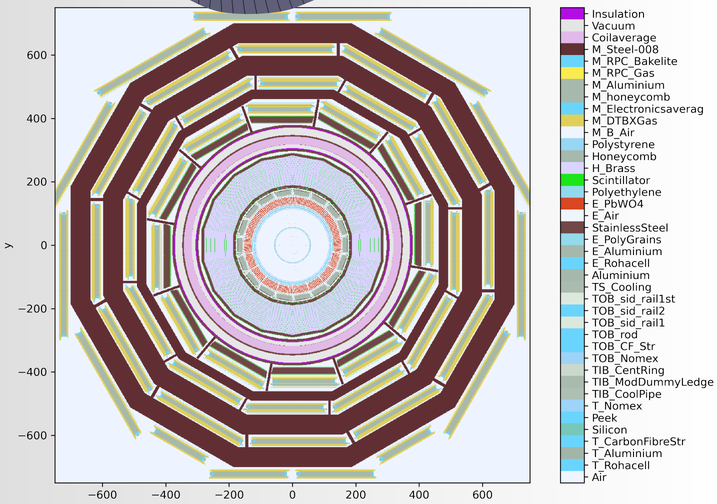
Celeritas is not designed to be a prototype code

Regression/timing suite

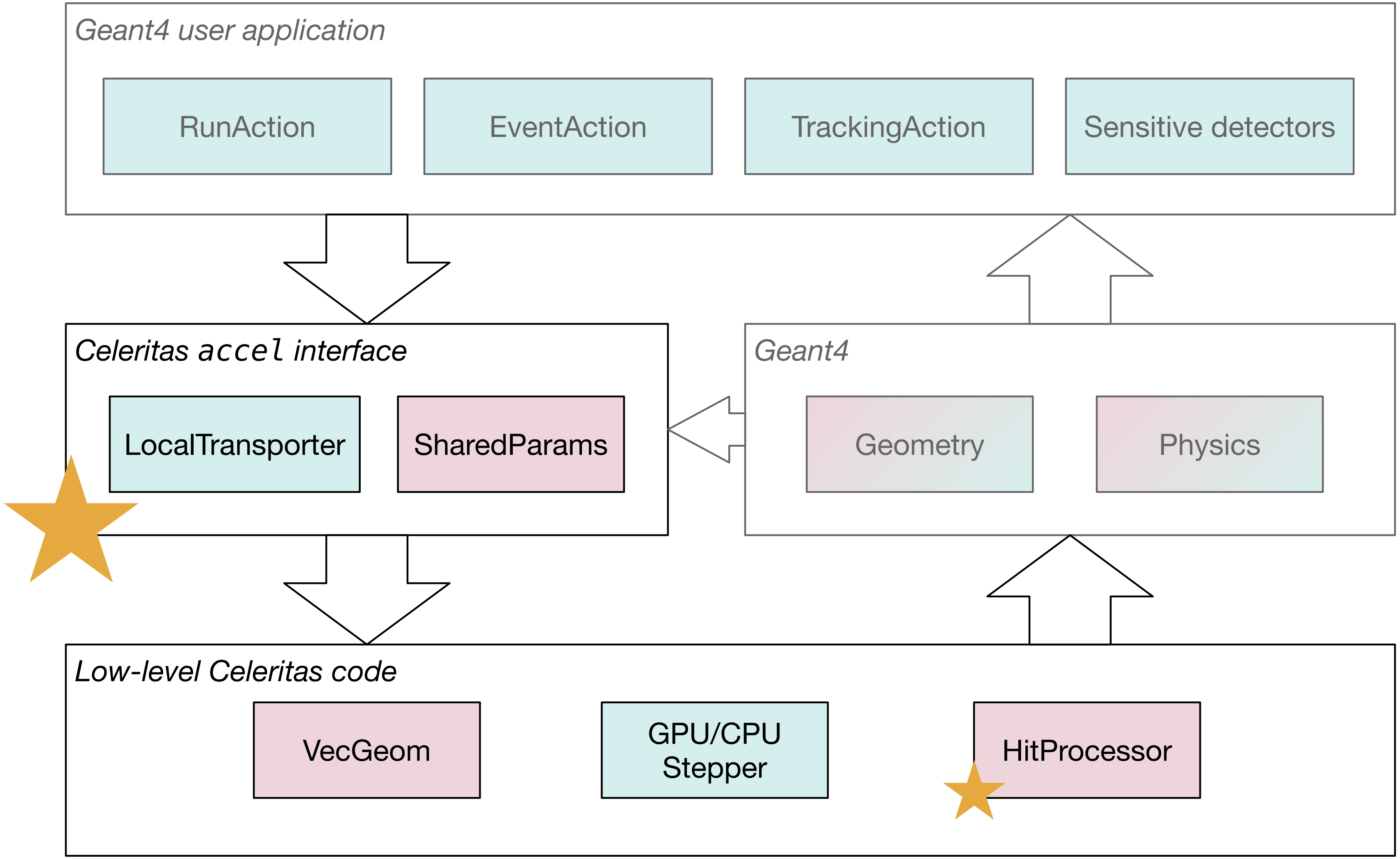
- Run on single node of Summit at full capacity
  - 6 separate runs simultaneously (different seed for each)
  - Each run: 7 CPU (OpenMP) vs. 1 GPU (+1 CPU)
  - Demonstrate performance “loss” by neglecting GPU resources
- 1300 10 GeV e^- per event, 7 events per run
- Preliminary set of problem definitions
  (working with AdePT team to develop)
- Initial optimizations
- Initial results are apples-to-apples







Geant4 interface library

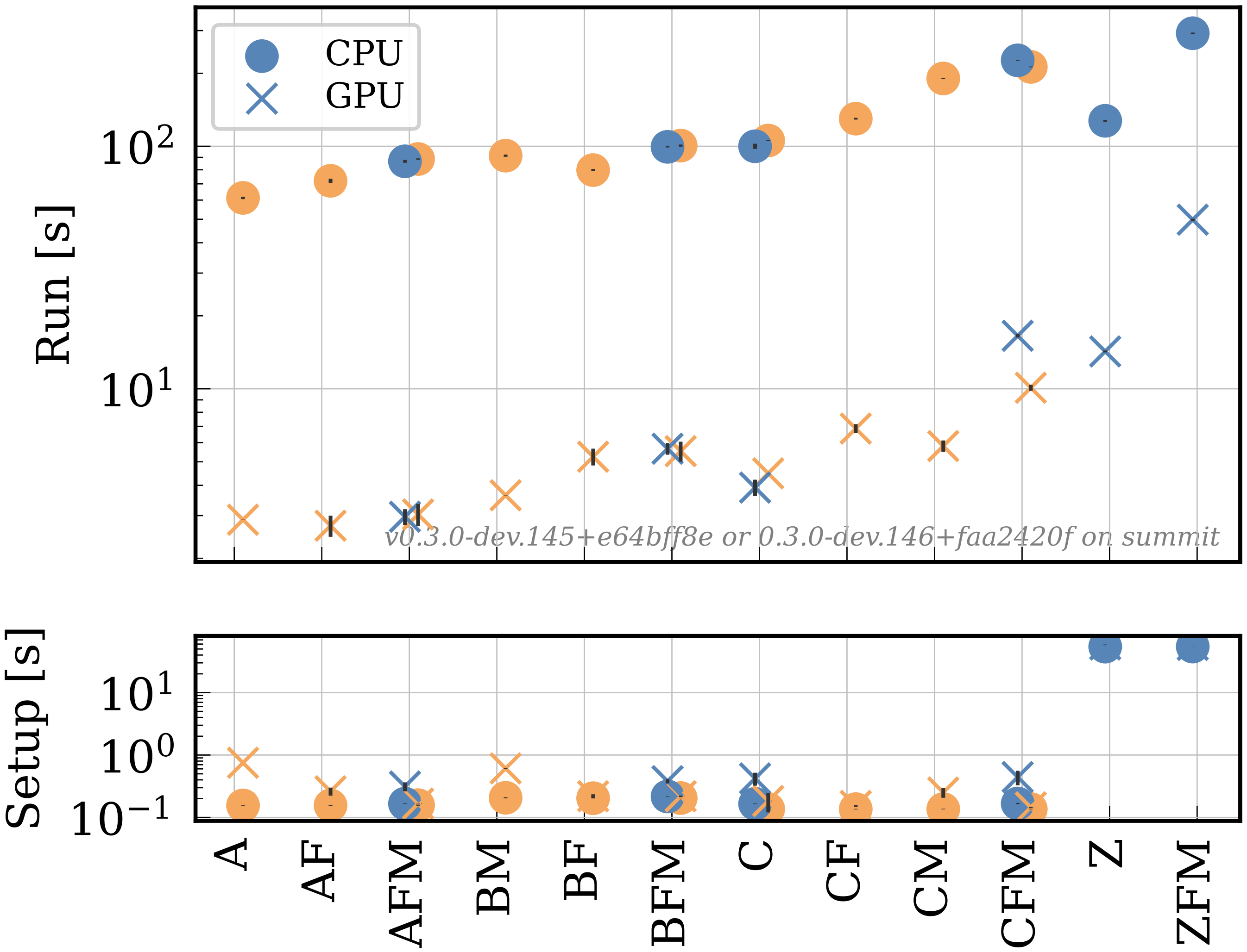


Thread-local
Shared

https://celeritas-project.github.io/celeritas/user/index.html

**New code for v0.2*

Regression problem run time



ORANGE
VecGeom

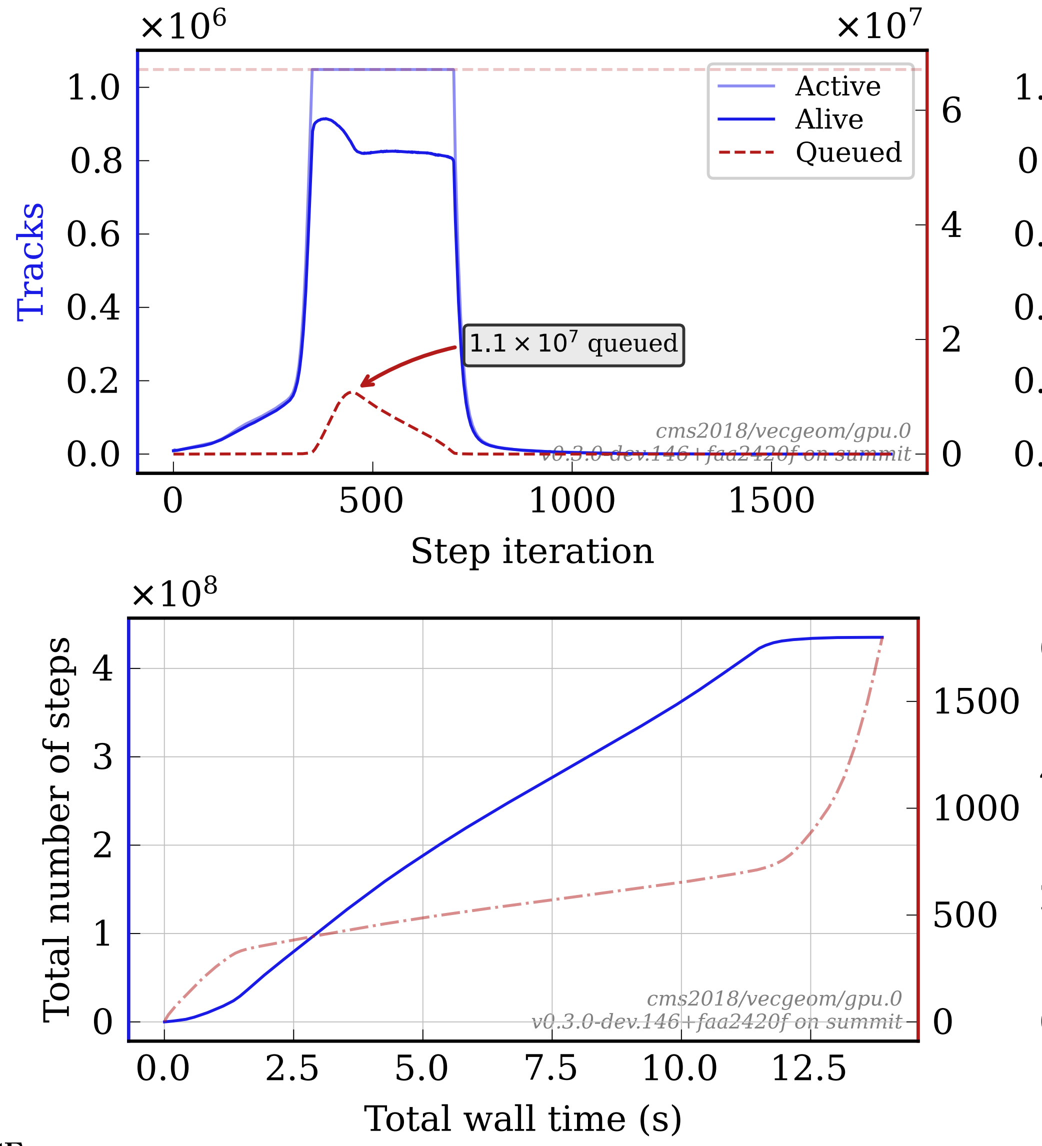
Problem definition

A	testem15
B	simple-cms
C	testem3
Z	cms2018

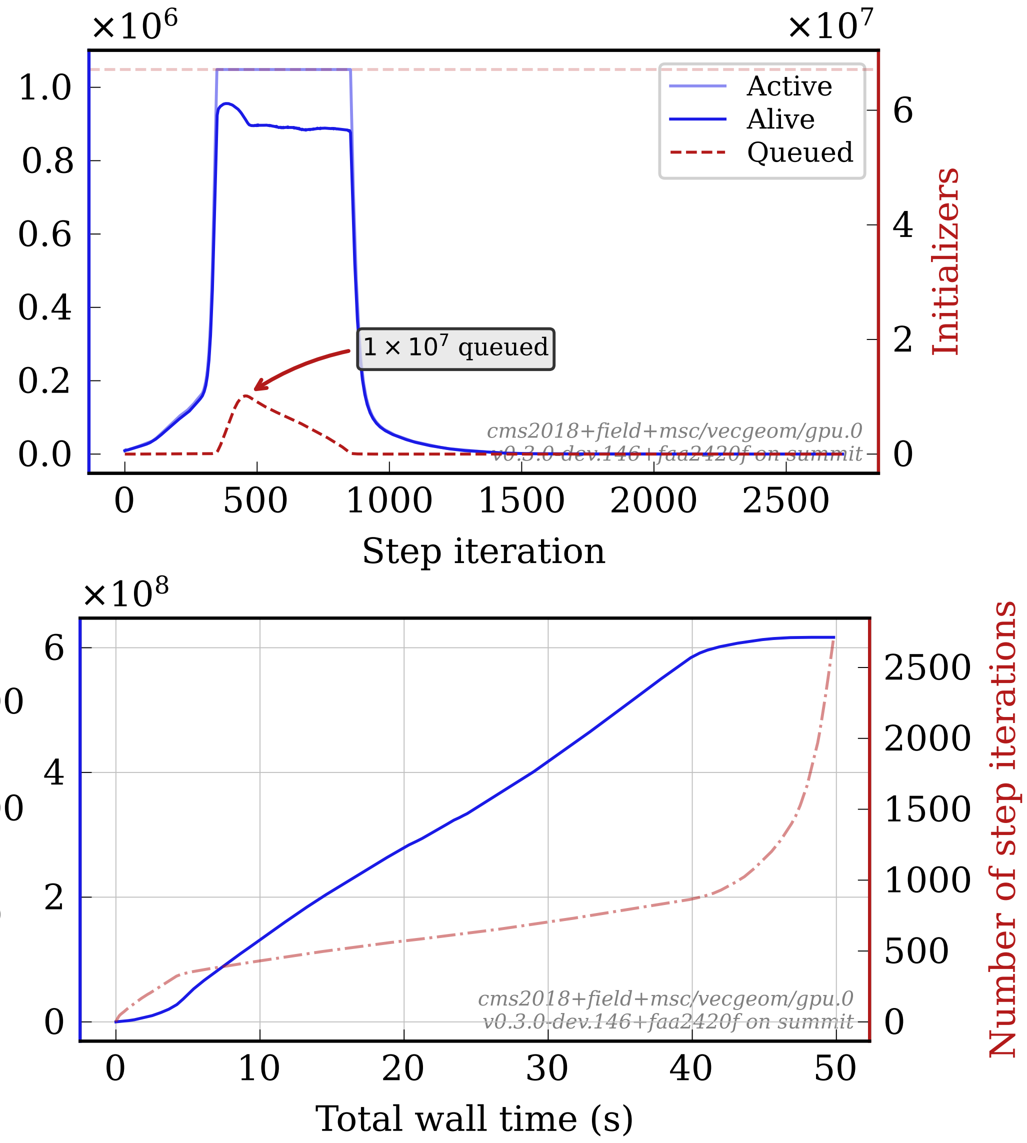
Modifier

F	+field
M	+msc

CMS2018 performance



CMS2018 ***GPU*** *without field or MSC*



CMS2018 ***GPU*** *with field and MSC*

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

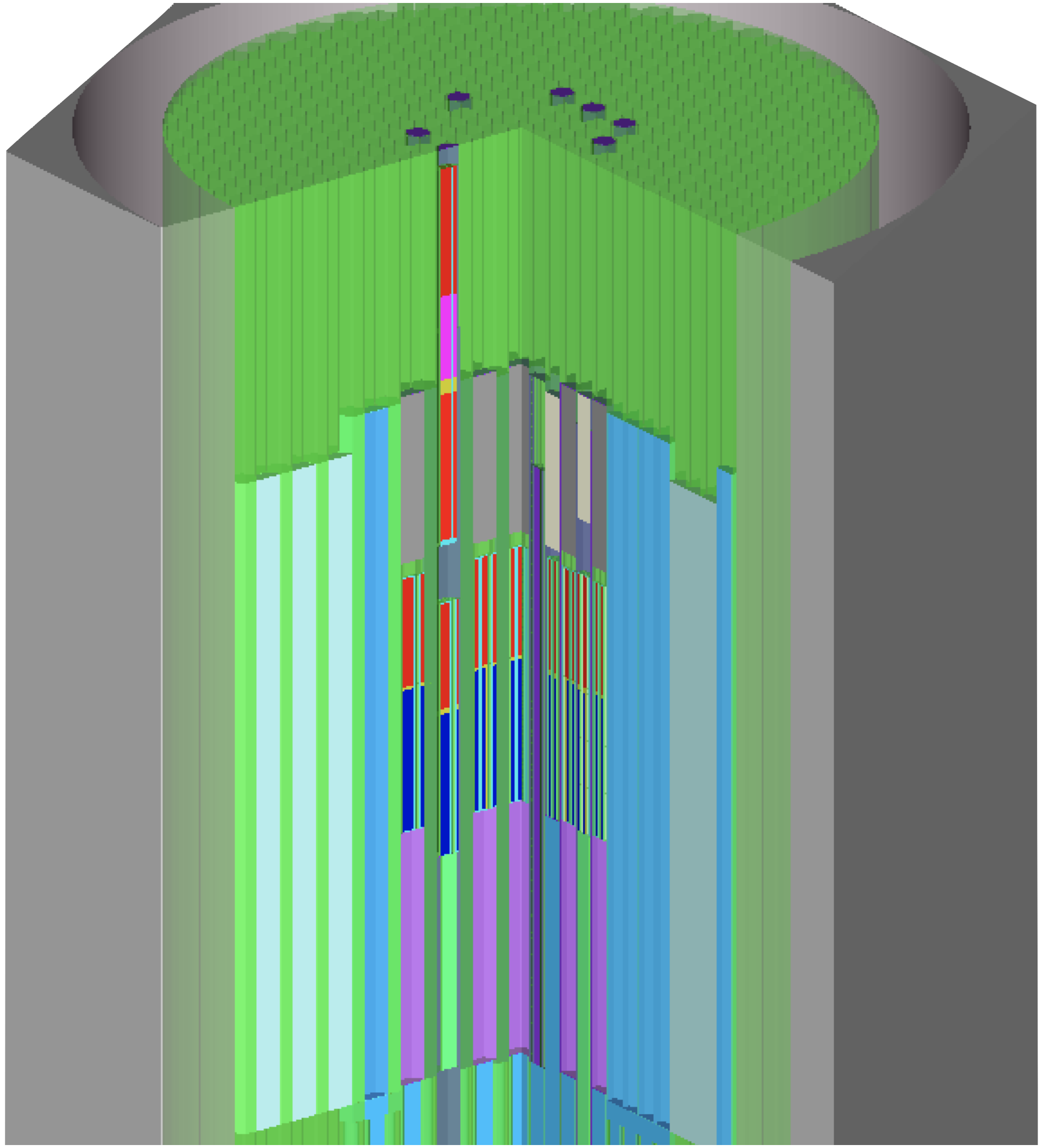
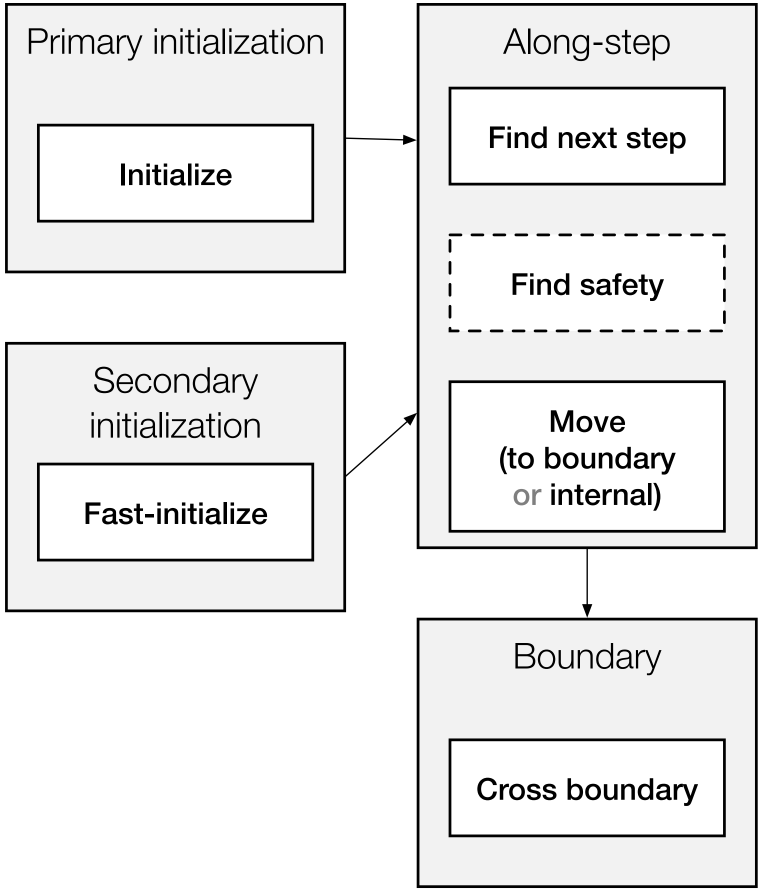


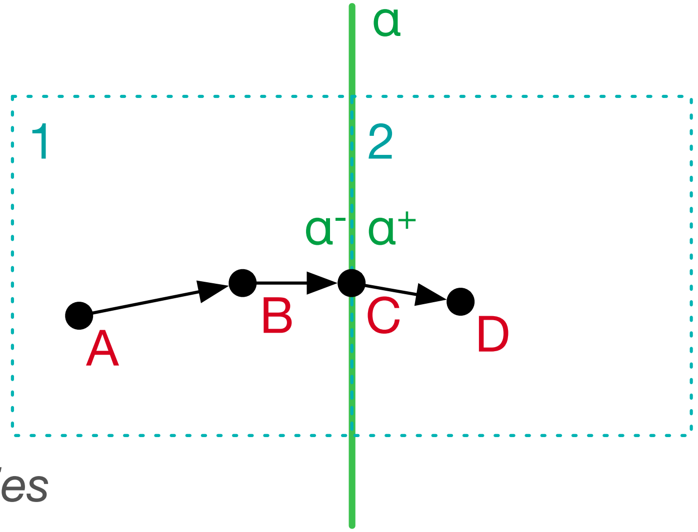
Image credit: Steve Skutnik (ORNL)

ORANGE surface-based tracking methodology

Celeritas geometry interface

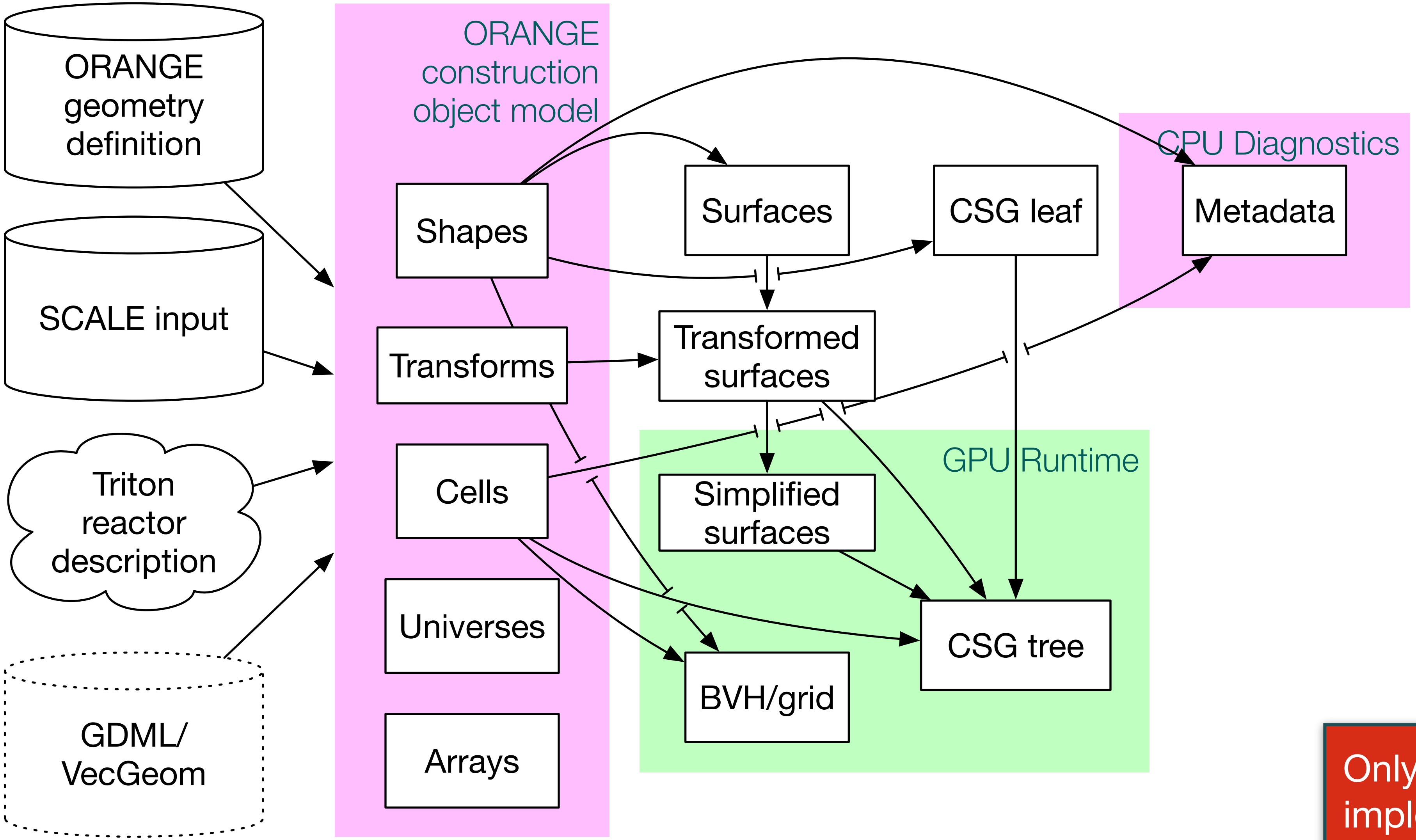


	Position	Volume	Surface+Sense
(input)	A	—	
Initialize	A	1	—
Find step	A	1	—
Move internal	B	1	—
Move to bdy	C	1	α inside
Cross bdy	C	2	α outside
Move internal	D	2	—



**exact handling of direction changes on boundaries*

ORANGE surface/volume construction



Only **partially** implemented in Celeritas ORANGE