# Celeritas: efficient detector simulation on GPUs for Geant4

Seth R Johnson

*Celeritas Code Lead*
*Senior R&D Staff*
*Scalable Engineering Applications*

*Celeritas core team:*

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

*Celeritas core advisors:*

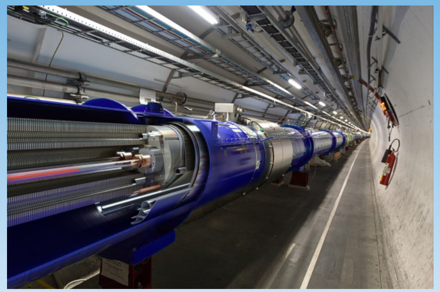
Tom Evans *(ORNL)*,
Philippe Canal *(FNAL)*,
Marcel Demarteau *(ORNL)*,
Paul Romano *(ANL)*

**Compute Accelerator Forum**
**11 December, 2023**

# Background
Methods
Results
Conclusions

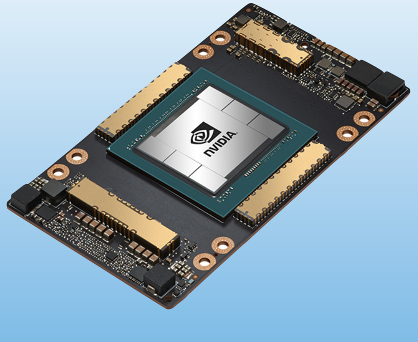
# Celeritas project goal

- **Accelerate scientific discovery** by improving LHC detector simulation throughput and energy efficiency
  - Long term goal: as much work as possible on GPU
  - Initial funding: focus on EM physics (but keep door open for more!)
- Jointly funded by US DOE ASCR and HEP
  - **Research and develop** novel algorithms for GPU-based Monte Carlo simulation in High Energy Physics
  - **Implement** production-quality code for GPU simulation
  - **Integrate** collaboratively into experiment frameworks



LHC beamline ©CERN

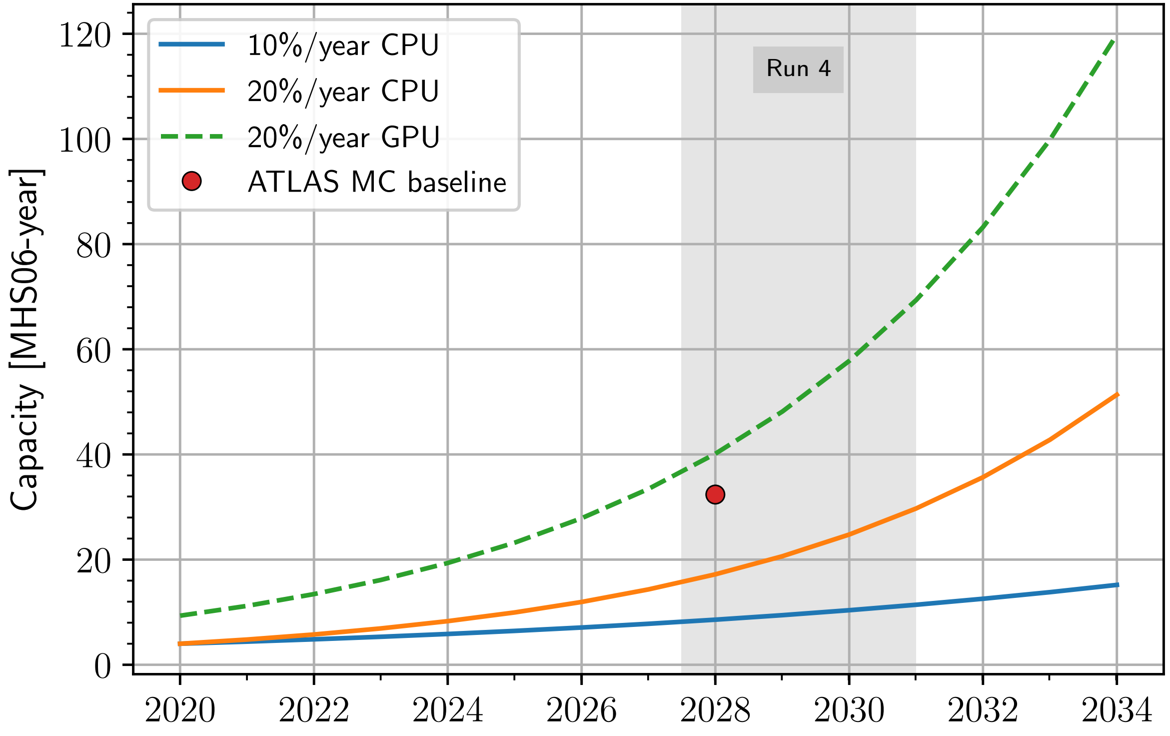
CELERITAS



Nvidia A100 GPU ©Nvidia

# Motivation 1: computational demand

- HL 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
- Even AI/ML based “fast simulation” methods will need lots of training data

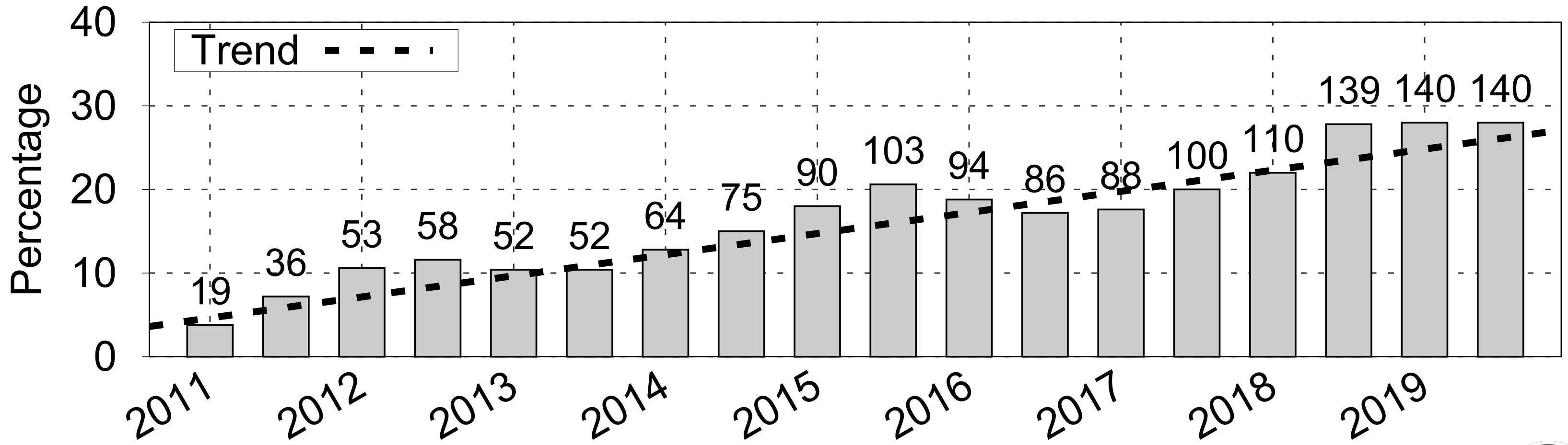



*GPU projection based on energy efficiency and speedup of ExaSMR MC code*

# Motivation 2: computational supply

- "Heterogeneous" architectures are increasingly common in high performance computing
- Scientific codes can run on GPU with much higher energy efficiency
  *e.g., Perlmutter reports 5× average: https://blogs.nvidia.com/blog/gpu-energy-efficiency-nersc/*
- Demand for AI/ML training and models ***will accelerate this trend***

Top500 supercomputers with heterogeneous architectures: >30%




Khan et al., *An Analysis of System Balance and Architectural Trends Based on Top500 Supercomputers*
Reproduced under government license. https://doi.org/10.1145/3432261.3432263

# ...but there's a catch

- Exascale Computing Project (ECP) funded a wide range of scientific libraries and applications to run efficiently on next-generation GPUs
- In *all* cases, performance on GPU requires:
  - Algorithmic restructuring *(reorganizing data, separating states, transposing loops)*
  - New numerical approaches *(targeting higher compute-to-memory ratios)*
  - Alternative physics models *(more favorable to thread-level parallelism)*
  - **Not simply porting code**

**Drastically different hardware** requires **dramatically different software**

Background

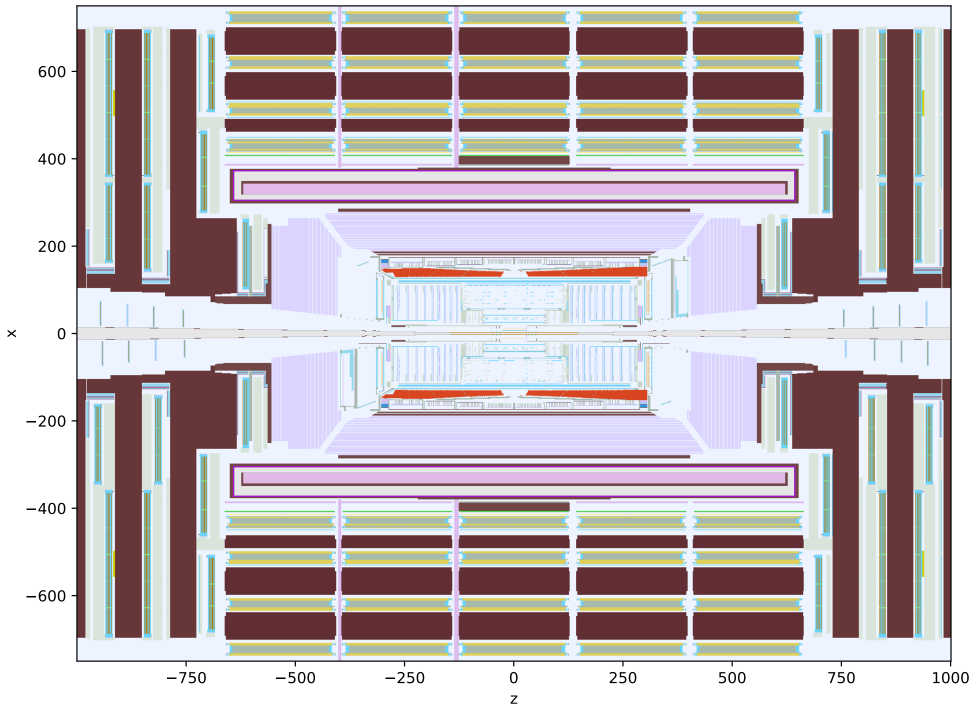
**Methods**

Results

Conclusions

# High-level capabilities targeting LHC simulation

- Equivalent to `G4EmStandardPhysics`
  *…using Urban MSC for high-E MSC; only γ, $e^{\pm}$*
- Full-featured Geant4 detector geometries using VecGeom 1.x
- Runtime selectable processes, physics options, field definition
- Execution on CUDA (Nvidia), HIP* (AMD), ***and CPU*** devices




*GPU-traced rasterization of CMS 2018*

**VecGeom currently requires CUDA: ORANGE navigation required for HIP*

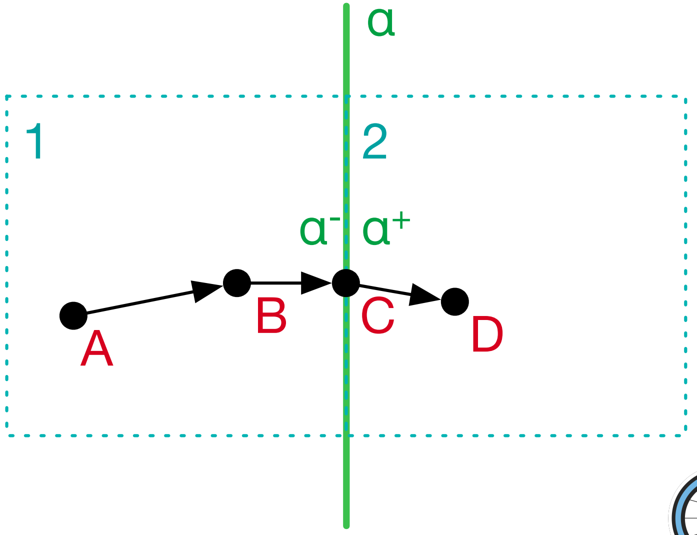
*Source:* Johnson, S.R. Geant4 Meeting 2023

# ORANGE: surface-based navigation

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

| | 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 | — |

*Discrete state points (avoiding "fuzziness") is optimal for GPU*



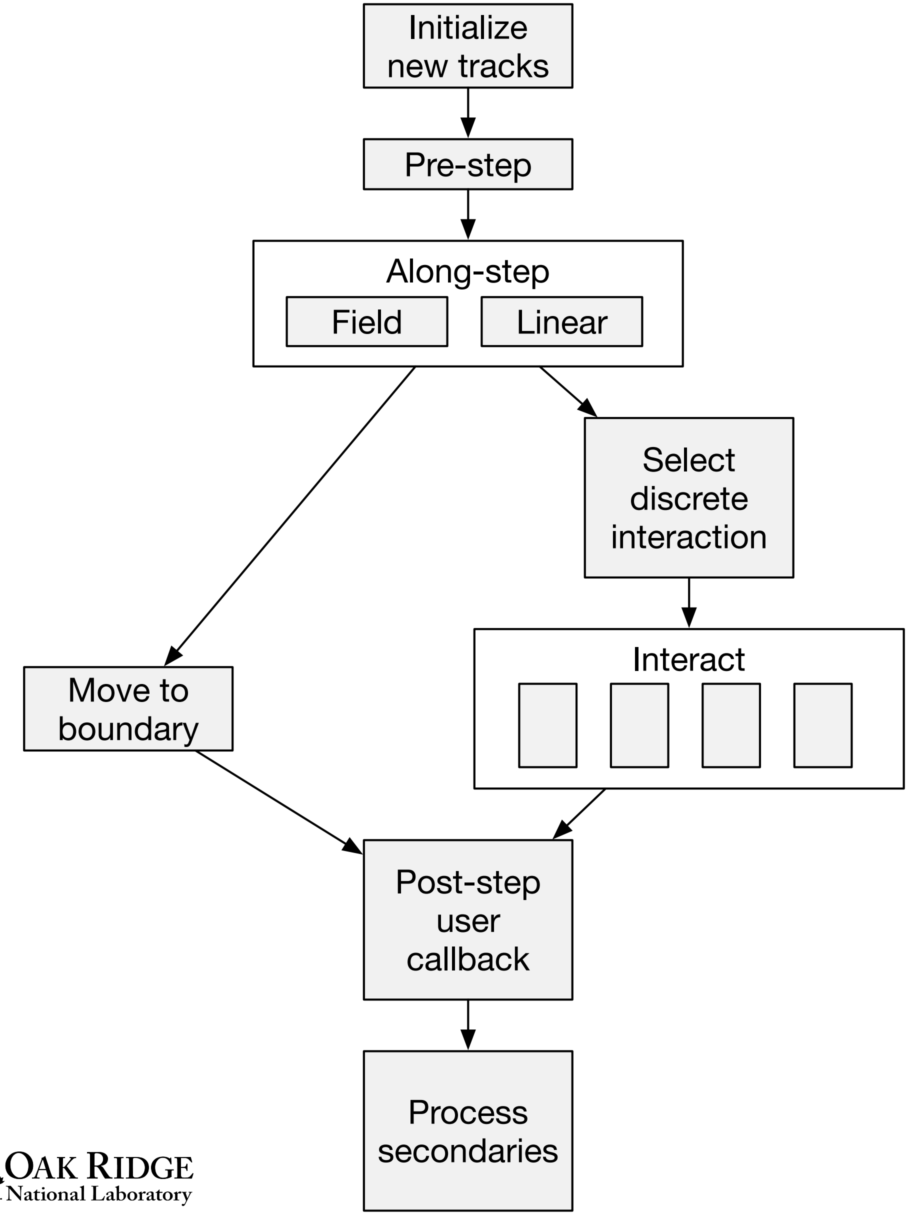
# Magnetic field propagation

- Composition based: **P○D○I○E○F**
- Templated for extensibility
  - Built-in "uniform" and "*r-z* field map"
  - Magnetic field (Lorentz) equation
  - Single driver (for now) with runtime step tolerances
  - Runge–Kutta 4 and Dormand–Prince RK5(4)7M integrators
  - Custom field propagator *without* safety evaluation*

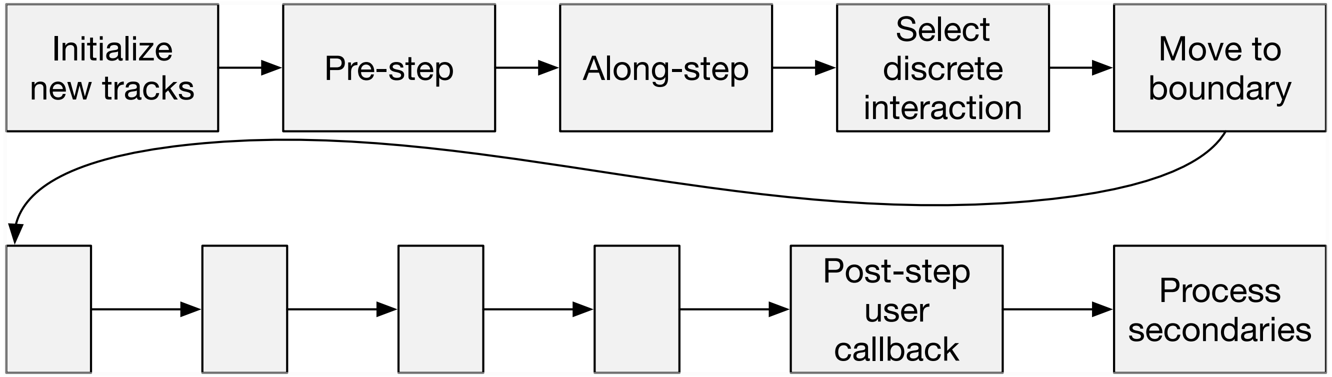
| Operator | Input | Output |
|---|---|---|
| **F**ield | $\boldsymbol{x}$ | $\boldsymbol{B}$ |
| **E**quation of motion | $\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{B}$ | $\boldsymbol{x}', \boldsymbol{p}'$ |
| **I**ntegrator | $\boldsymbol{x}, \boldsymbol{p}, h$ | $\boldsymbol{x}^*, \boldsymbol{p}^*, e$ |
| **D**river | $\boldsymbol{x}, \boldsymbol{p}, s$ | $\boldsymbol{x}^*, \boldsymbol{p}^*, s^*$ |
| **P**ropagator | $\boldsymbol{x}, \boldsymbol{\Omega}, E, s$ | $\boldsymbol{x}^*, \boldsymbol{\Omega}^*, s^*$ |

**safety calculation resulted consistently in slowdown on GPU*

# Stepping loop on a GPU




*Process large batches of tracks per kernel ($10^3$–$10^6$)*




*Topological sort: a loop over kernels*

*Using many small kernels improves extensibility*





*Source:* Johnson, S.R. Geant4 Meeting 2023

# Celeritas/Geant4 integration

- **Imports** EM physics selection, cross sections, parameters
- **Converts** geometry to VecGeom model without I/O
- **Offloads** EM tracks from Geant4

  *(Via `G4UserTrackingAction`, `G4VFastSimulationModel`, or `G4VTrackingManager`)*
- **Scores** hits to user "sensitive detectors"

  *(Copies from GPU to CPU; reconstructs `G4Hit`, `G4Step`, `G4Track`; calls `Hit`)*
- **Builds** against Geant4 10.5–11.1

***Celeritas has production quality interfaces to simplify user application integration***




*Source:* Johnson, S.R. Geant4 Meeting 2023

Background

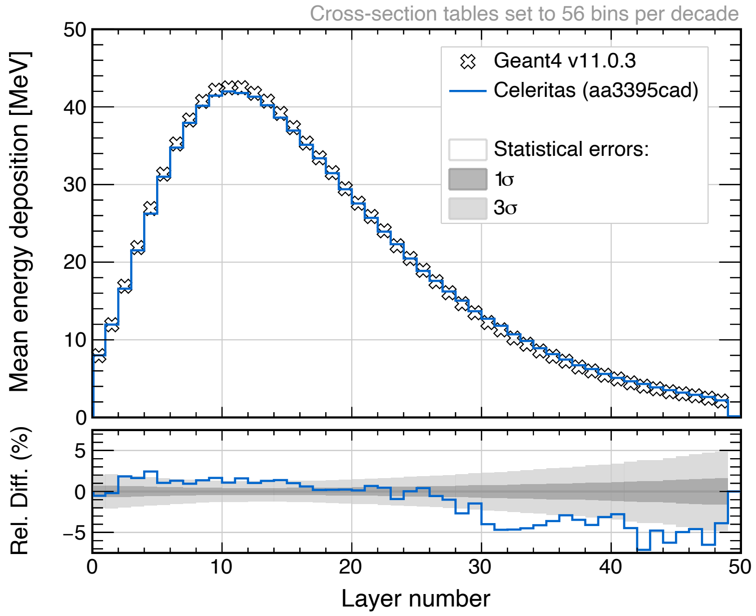
Methods

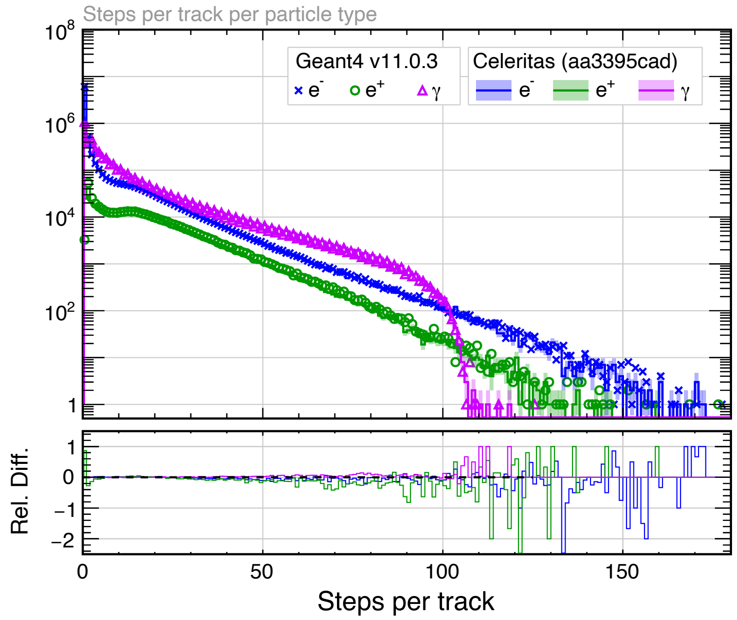
**Results**

Conclusions

# Physics verification

- Single-model distributions
- Volume-dependent hit count and energy deposition distributions
- Step-per-track distributions
- Most significant disagreement remaining: Urban MSC



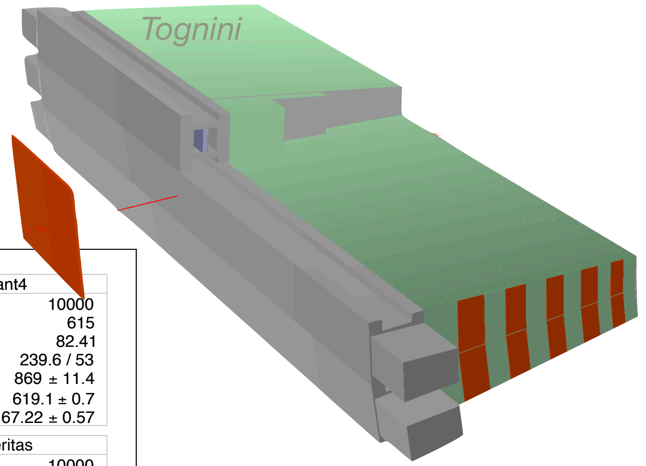



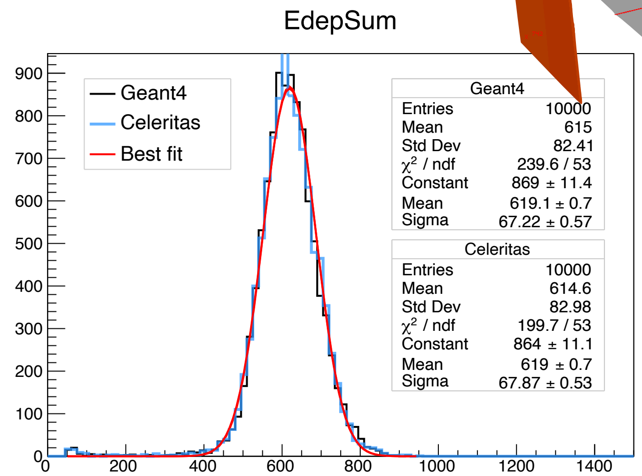


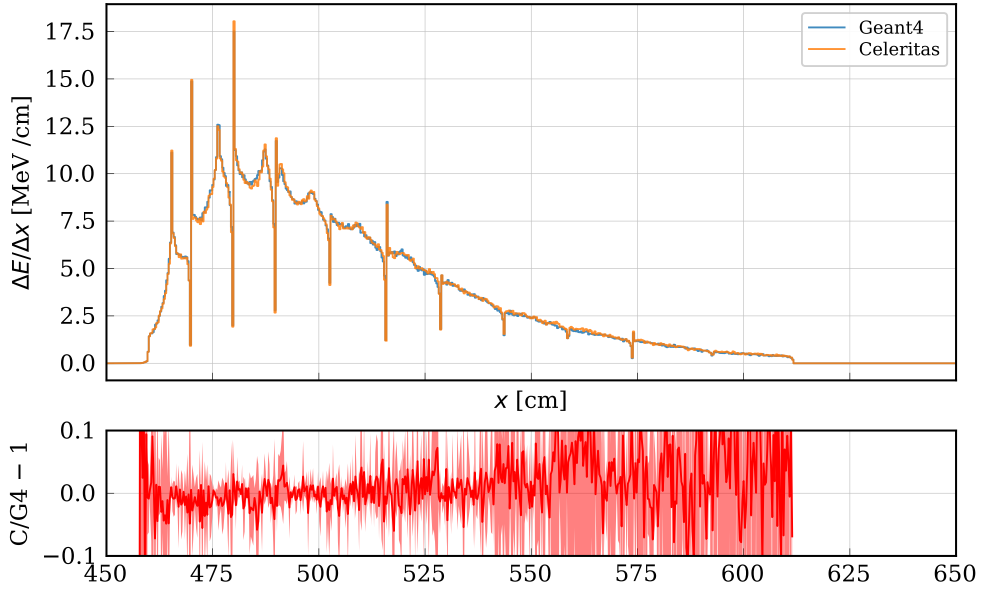
OAK RIDGE National Laboratory

# EM offloading with FullSimLight

- ATLAS FullSimLight: hadronic tile calorimeter module segment
  - 64 segments in full ATLAS, 2 in this test beam
  - 18 GeV $\pi^+$ beam, no field
  - FTFP_BERT (default) physics list
    (includes standard EM)
- **~100 lines of code to integrate**
  - Offload $e^-$, $e^+$, $\gamma$ to Celeritas
  - Celeritas reconstructs hits and sends to user-defined `G4VSensitiveDetector`
- Good agreement in energy deposition




*yz-integrated energy deposition*

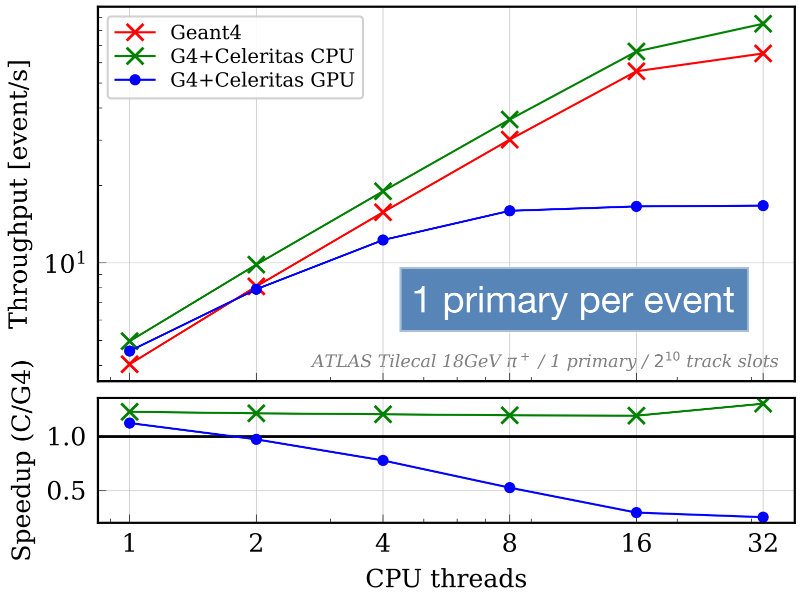
*Test problem: Lachnit, Pezzotti; FSL integration: Morgan*

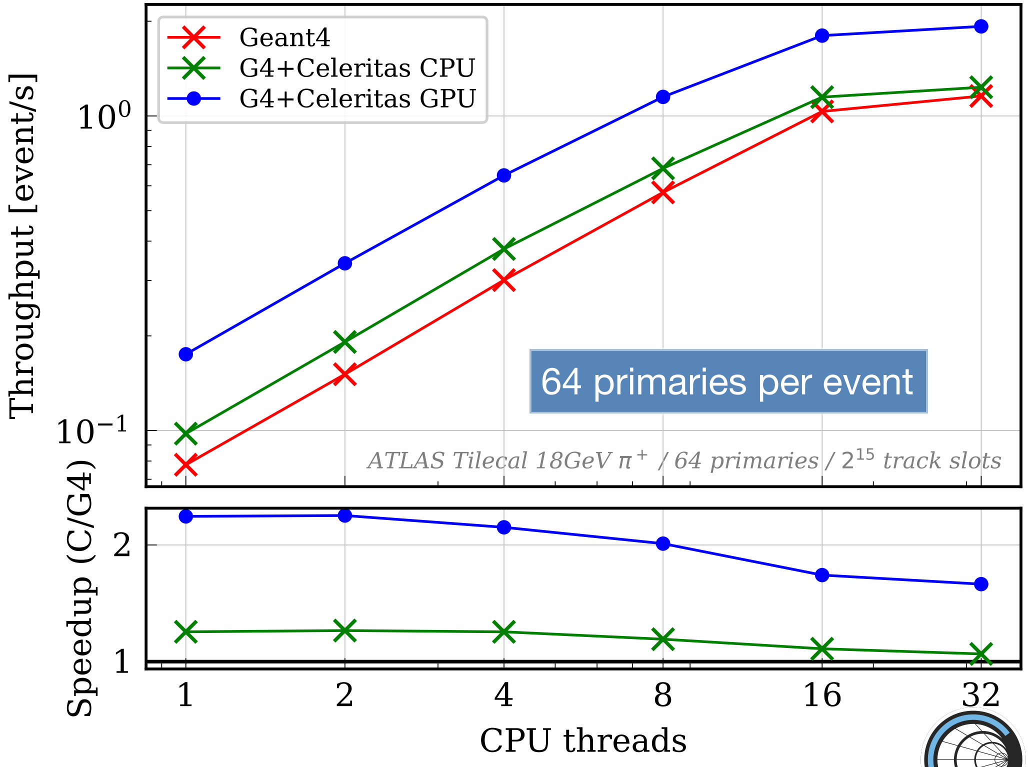
*Source:* Johnson, S.R. Geant4 Meeting 2023

Oak Ridge National Laboratory

# Offload performance results

- 1/4 of a Perlmutter (NERSC) GPU node
  16 cores of AMD EPYC, 1 Nvidia A100
- Time **includes** startup overhead, Geant4 hadronic physics, track reconstruction, and SD callback
- GPU speedup: **1.7–1.9×** at full occupancy
  Using all CPU cores with a single GPU
- CPU-only speedup: **1.1–1.3×**
- LHC-scale energy per event (i.e., all 64 modules) is needed for GPU efficiency
- One fast GPU can be shared effectively by full multithreaded Geant4

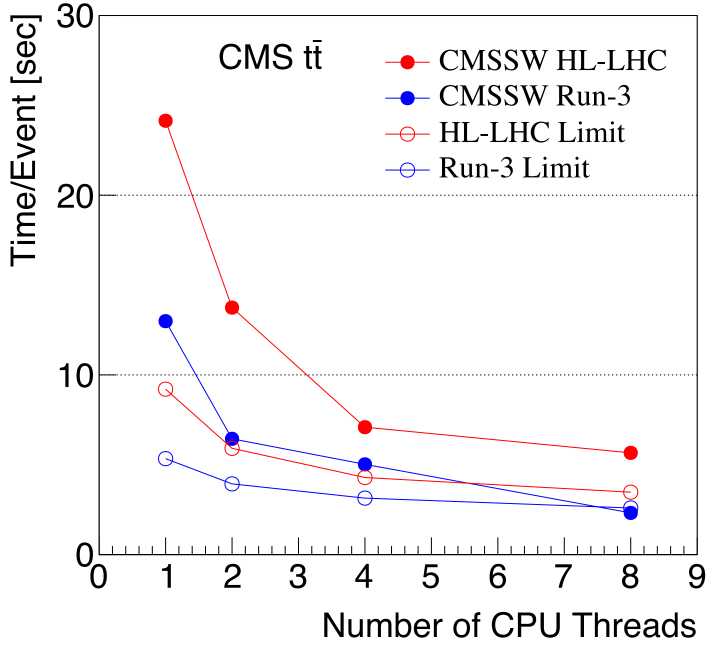


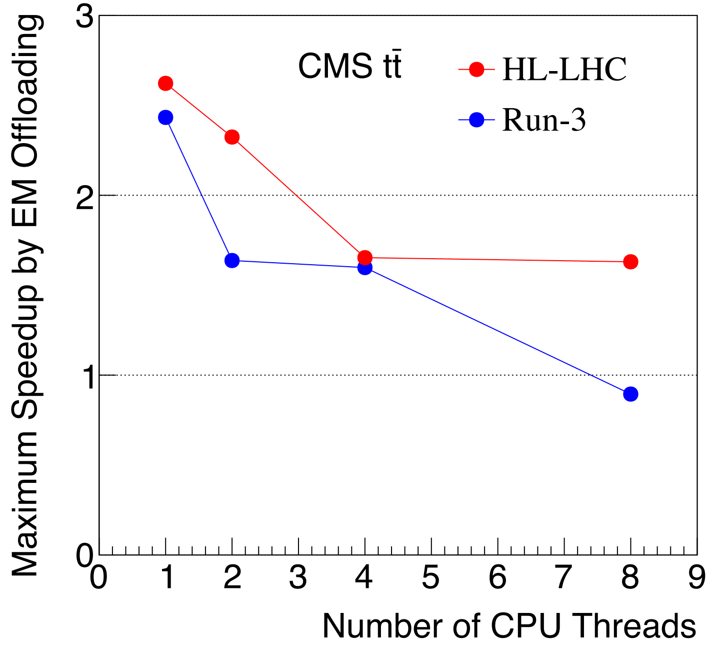




Source: Johnson, S.R. Geant4 Meeting 2023

# CMSSW integration

- Initial CMSSW integration complete
  - 500 lines of code
  - Complications from extra user track state
- Performance isn't comparable due to different physics
  - Lots of region-dependent cuts, parameter changes
  - Fast simulation bypasses transport loop
- Strong collaboration with CMS
  - **CMSSW has agreed** to integrate Celeritas as an external
  - CPU-only for now to facility software infrastructure
  - Maximum speedup for offloading EM: ~2.5×



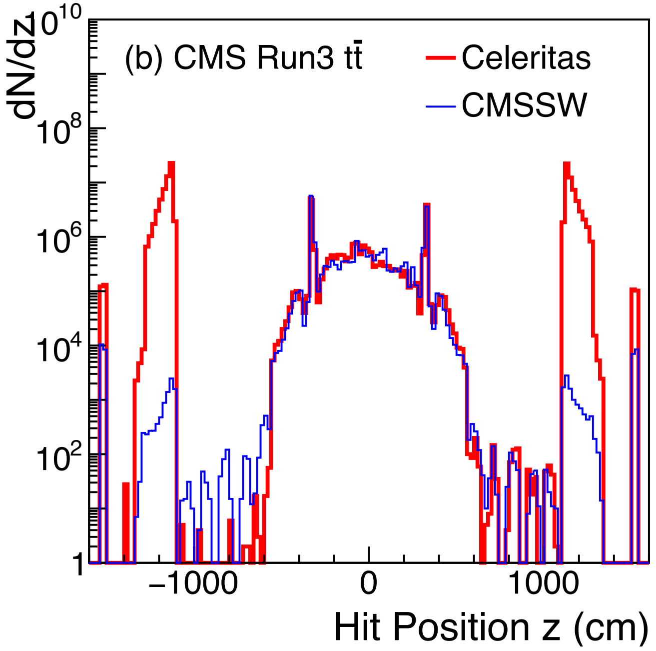



# CMS Run 3&4 Standalone Simulations

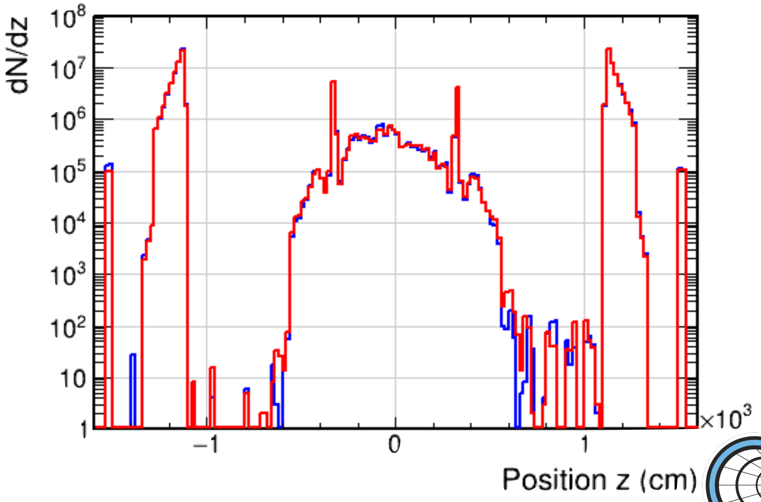
- Standalone Geant4 app `celer-g4`
- 32 $t\bar{t}$ events from Pythia
- FTFP_BERT physics
  - Geant4 simulates hadronics
  - All EM tracks offloaded to Celeritas
  - Lepto-nuclear reactions neglected
- Multiple field options
  - No magnetic field
  - Uniform 4T field
  - Discretized+interpolated RZ field (901×481 points)
- CMSSW/Geant4 throughput: **8×**

  *(we're simulating a harder problem than necessary, but we now have an equivalent test problem)*

*CMSSW Run 3 hit distribution*

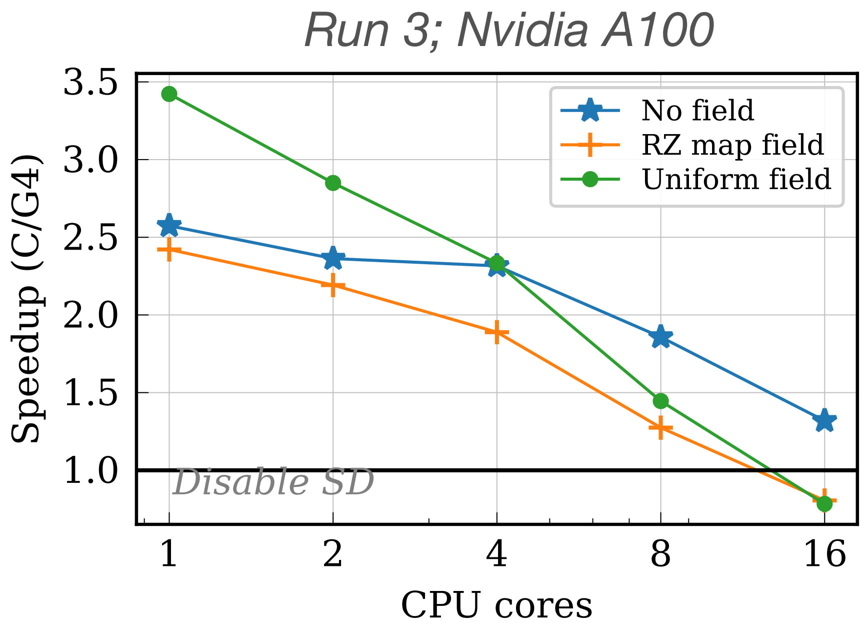


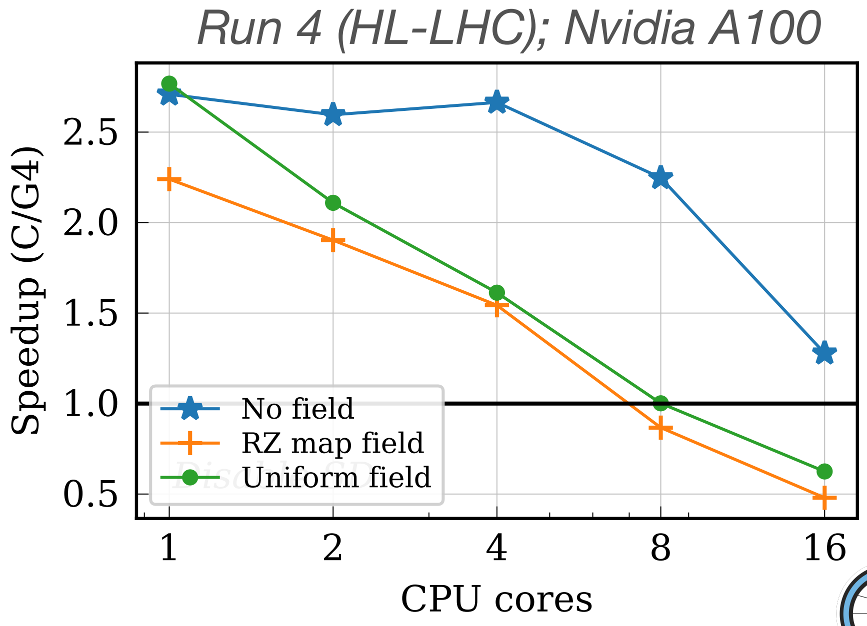

*Standalone Run 3 hit distribution*



# CMS Run 3&4 Standalone Results

- Promising performance
  - SD reconstruction adds <15% overhead
  - Initial comparison of hits shows good agreement
  - Run 3: 25%–190% improvement at 8 cores
  - With task-based framework we might see better (due to less GPU contention)
- Possible future improvements:
  - Magnetic field propagation
  - Activating track sorting to get smaller kernel grid sizes
  - Single-precision? (Especially on consumer cards)



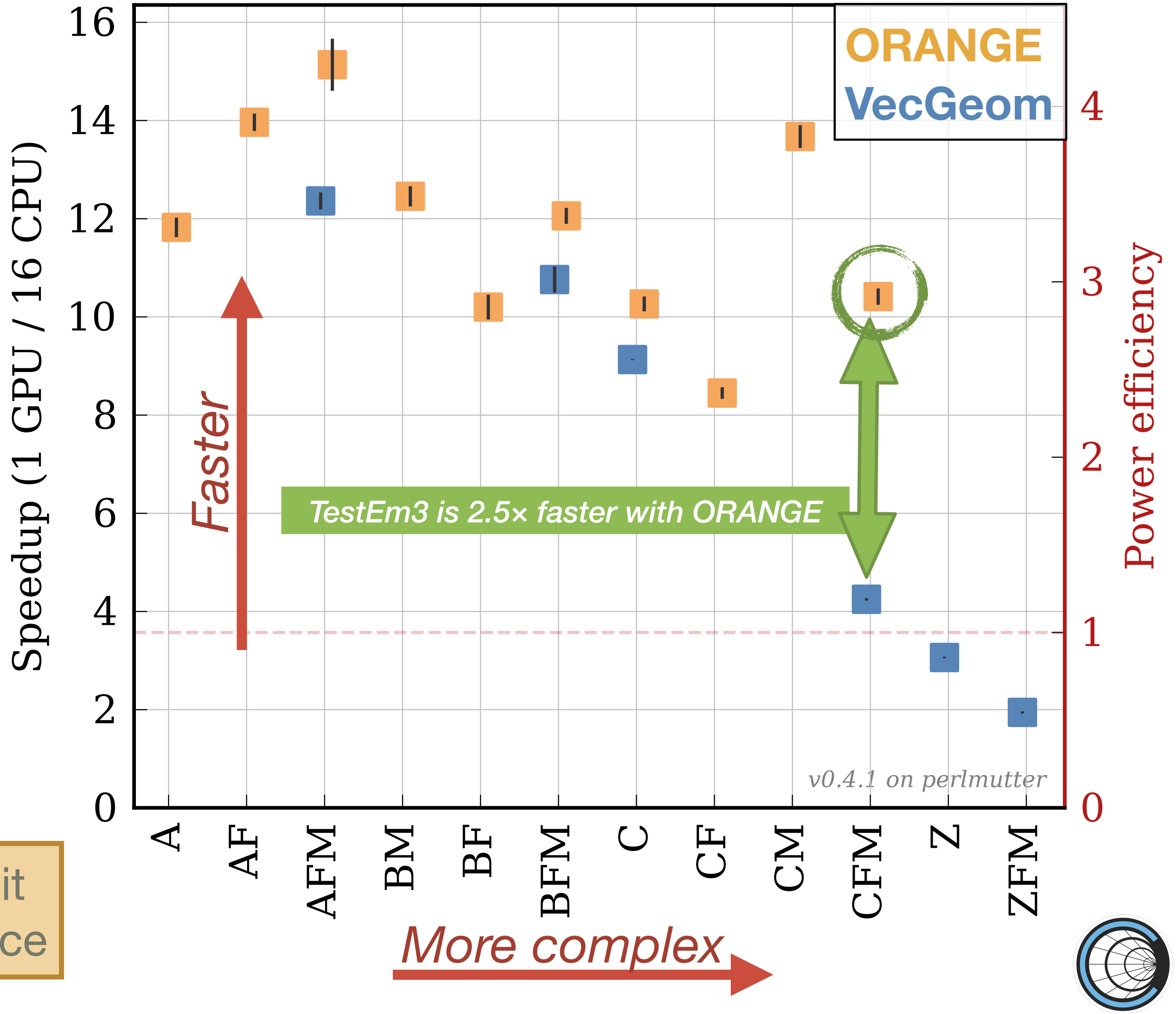

Nvidia A100 vs AMD 7532 EPYC

# Standalone EM performance

| | Problem definition |
|---|---|
| A | “infinite” medium |
| B | simple-cms |
| C | idealized calorimeter |
| Z | cms2018 |

| | Modifier |
|---|---|
| F | +field |
| M | +msc |

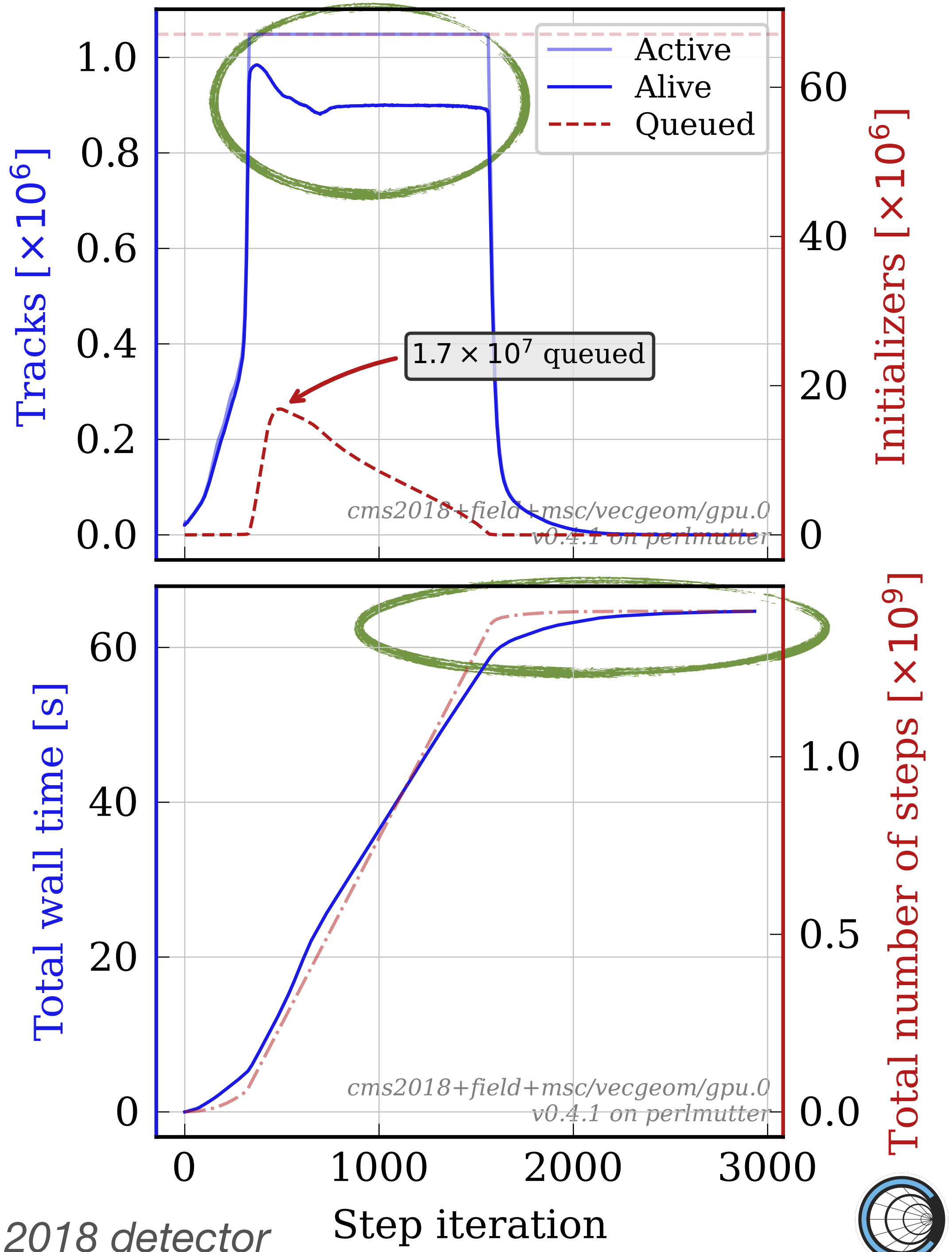
- 1300 × 10 GeV e-, 16 events
- ¼ Perlmutter node (NERSC)
  *1 × Nvidia A100 GPU, ¼ × 64-core AMD EPYC 7763*
- Celeritas GPU vs CPU
  *CUDA (1 CPU thread) vs OpenMP (16 CPU threads)*
- Key metrics favor GPU
  - *Capacity:* **50–94% loss** if GPUs are ignored
  - *Efficiency:* up to **4×** performance per watt




Previous versions of this slide used Summit which shows much worse CPU performance

# Step-dependent behavior

- Number of active particle tracks changes drastically due to EM shower
- Saturated GPU takes the most time but <50% of step iterations
  *Despite using masking instead of sorting!*
- Converting the tail of long-lived tracks does *not* kill us




*7 × 1300 × 10 GeV e⁻: CMS 2018 detector*

# Speedup with respect to Geant4

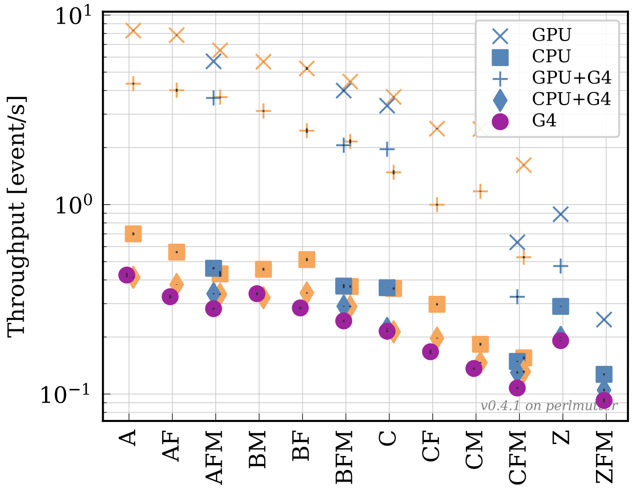
- Standalone Celeritas on CPU is ~50% faster than Geant4 for EM test suite
- GPU/G4 throughput: **2.5–20×**
- Still investigating disparity between "+G4" (offloaded from Geant4) versus standalone app

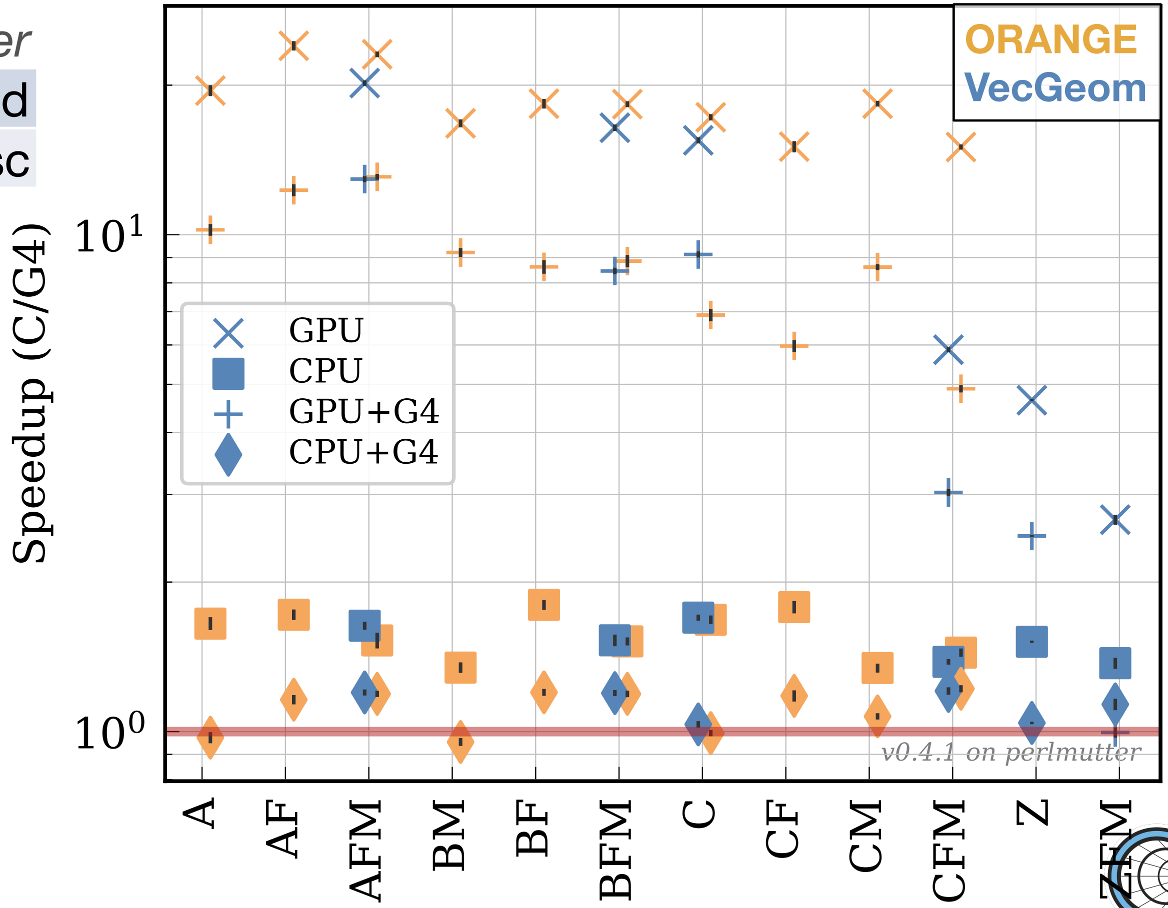
*Problem definition*

| | |
|---|---|
| A | "infinite" medium |
| B | simple-cms |
| C | idealized calorimeter |
| Z | cms2018 |

*Modifier*

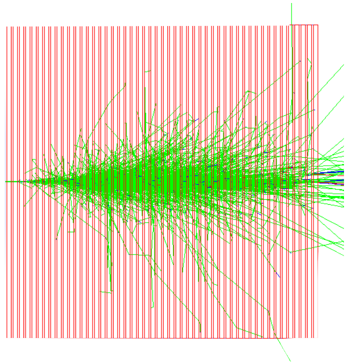
| | |
|---|---|
| F | +field |
| M | +msc |

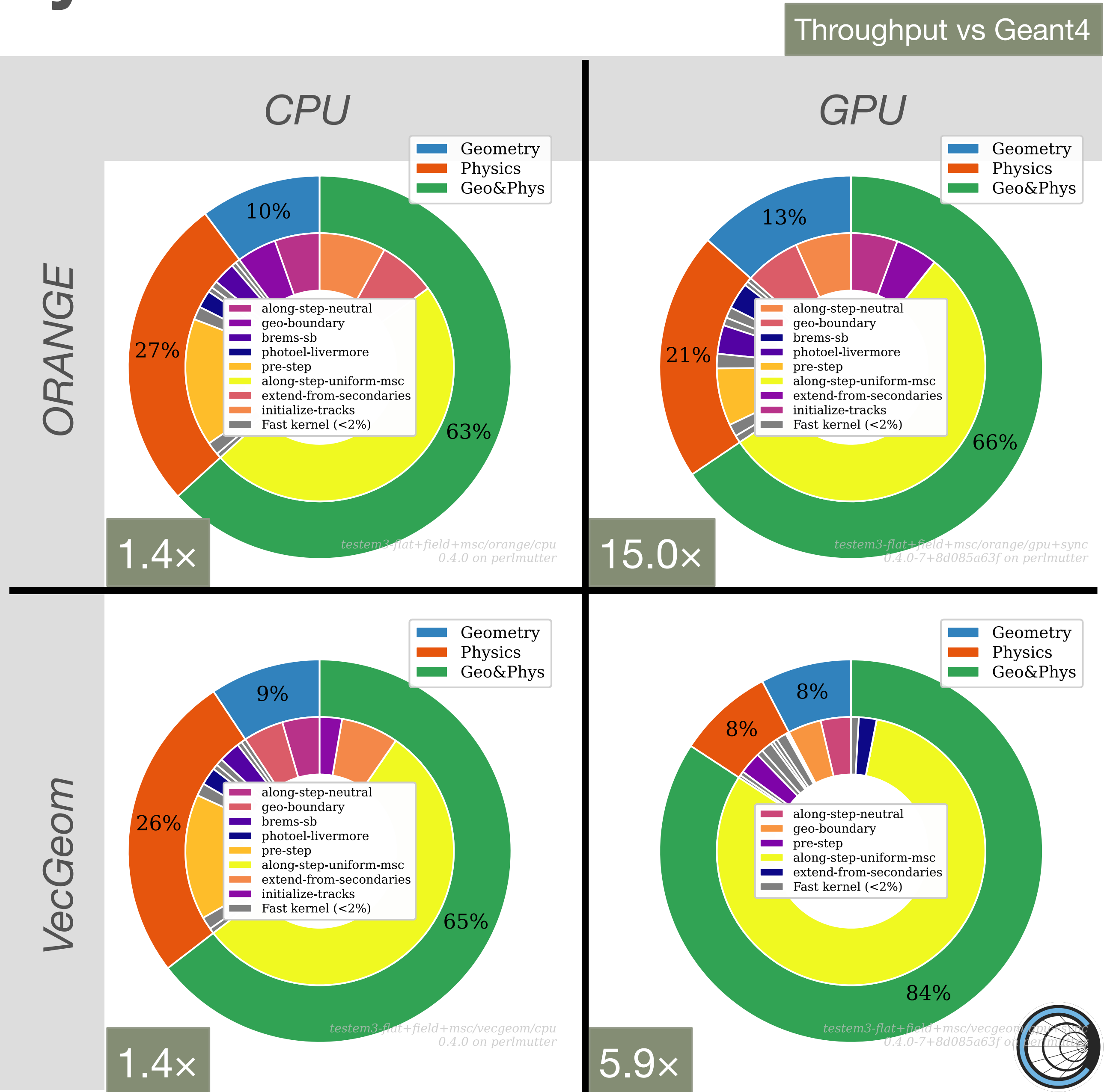


# TestEM3 performance disparity

- “No” divergence (all boxes)
- Performance parity on CPU
- *Physics* time parity on GPU
- Step counts are equivalent
- ORANGE faster on GPU
  - Neutral propagation: 1.4×
  - Field propagation: 3.6×
  - Boundary crossing: 1.5×



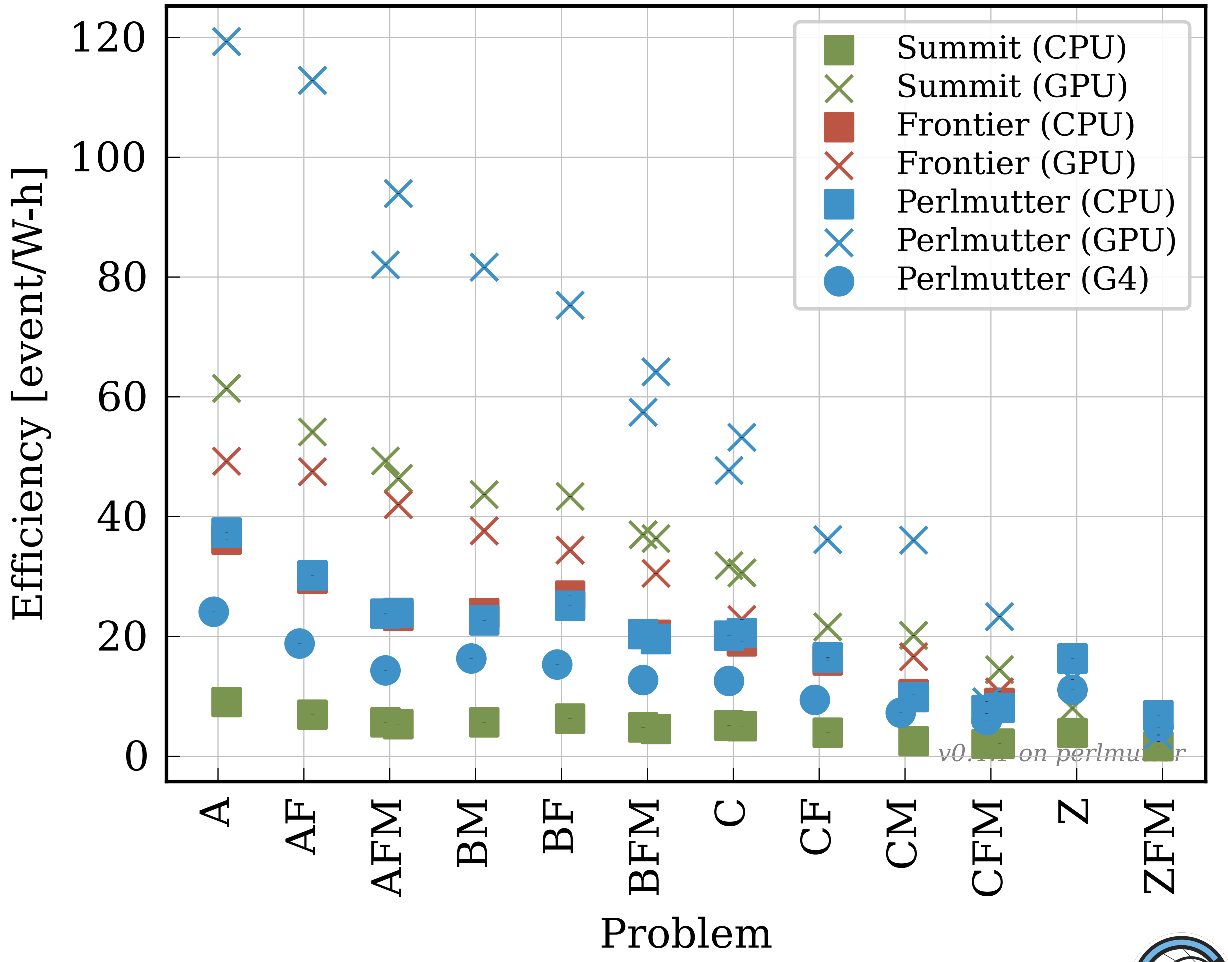



Oak Ridge National Laboratory

# Power efficiency

| Machine | GPU | CPU |
| --- | --- | --- |
| Summit | Nvidia V100 | IBM Power9 |
| Perlmutter | Nvidia A100 | AMD EPYC 7763 |
| Frontier | AMD MI250x | AMD EPYC 7453 |

- Estimated using reported Thermal Design Power (TDP) and Celeritas throughput
- GPU consistently shows higher energy efficiency 🌱
- A100/EPYC price: ~4× 💸




*EM only, no SDs*

Background

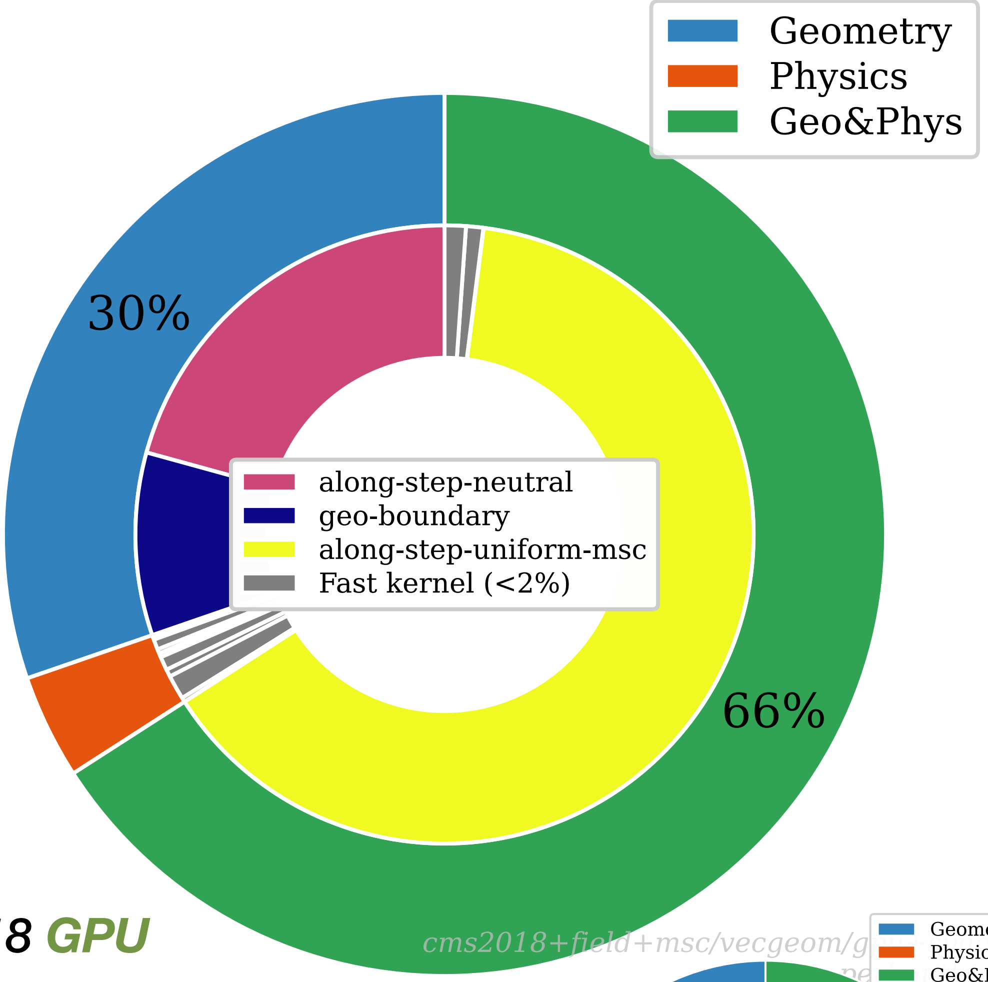
Methods

Results

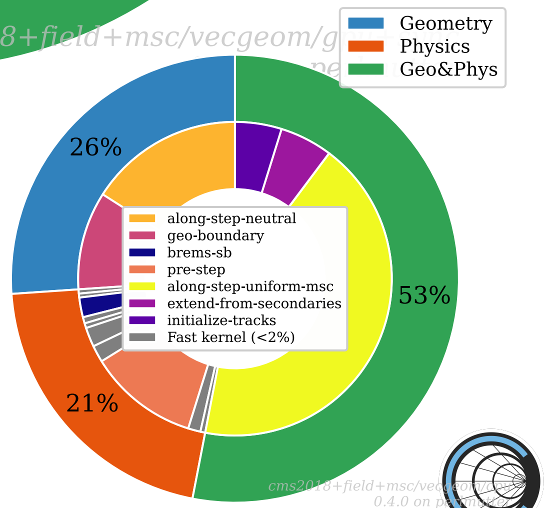
**Conclusions**

# Ongoing work

- Collaboration & integration
  - CMSSW
  - Athena (ATLAS) framework
  - Student projects
- Verification & validation
  - EM test problems
  - CMSSW workflow
- Optimization and geometry
  - **96%** of standalone runtime in CMS2018 is in geometry routines
  - GPU native sensitive detectors
  - ORANGE navigation
  - Track sorting




*CMS2018 GPU*




*CMS2018 CPU*



Oak Ridge National Laboratory

# Celeritas future

- Designed for **easy integration**
  - Potentially incorporate into Geant4 as an accelerator for certain HEP applications
  - Continue integration into HEP experiment frameworks
- Designed for **extensibility**
  - Optical photon simulation to be funded starting next year
  - Incremental addition of HEP physics for GPU offloading
- Designed for **performance**
  - Still have many avenues to investigate (without change to external interface!)
  - Surface-based geometry predicted to be much faster for complex applications
  - Works well on CPU, better on GPU

# Summary: by the numbers

| | | |
|---|---|---|
| **100** | **lines of code** | to integrate Celeritas into a FullSimLight tile calorimeter test application, with no modifications to Geant4 |
| **1.8×** | **full-simulation speedup** | including hadronics and SD hits, by using 1 Nvidia A100 with 16 AMD EPYC cores for the ATLAS test beam application *[NERSC Perlmutter]* |
| **2–20×** | **throughput** | when using Celeritas on GPU (compared to Geant4 MT CPU) for EM test problems *[NERSC Perlmutter]* |
| **4×** | **performance per watt** | for TestEM3 (ORANGE geometry) using Celeritas GPU instead of Geant4 CPU *[NERSC Perlmutter]* |

*Celeritas v0.4: https://celeritas-project.github.io/celeritas/*

# Acknowledgments

*Celeritas v0.4 code contributors:*

- Elliott Biondo (@elliottbiondo)
- Philippe Canal (@pcanal)
- Julien Esseiva (@esseivaju)
- Tom Evans (@tmdelellis)
- 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.

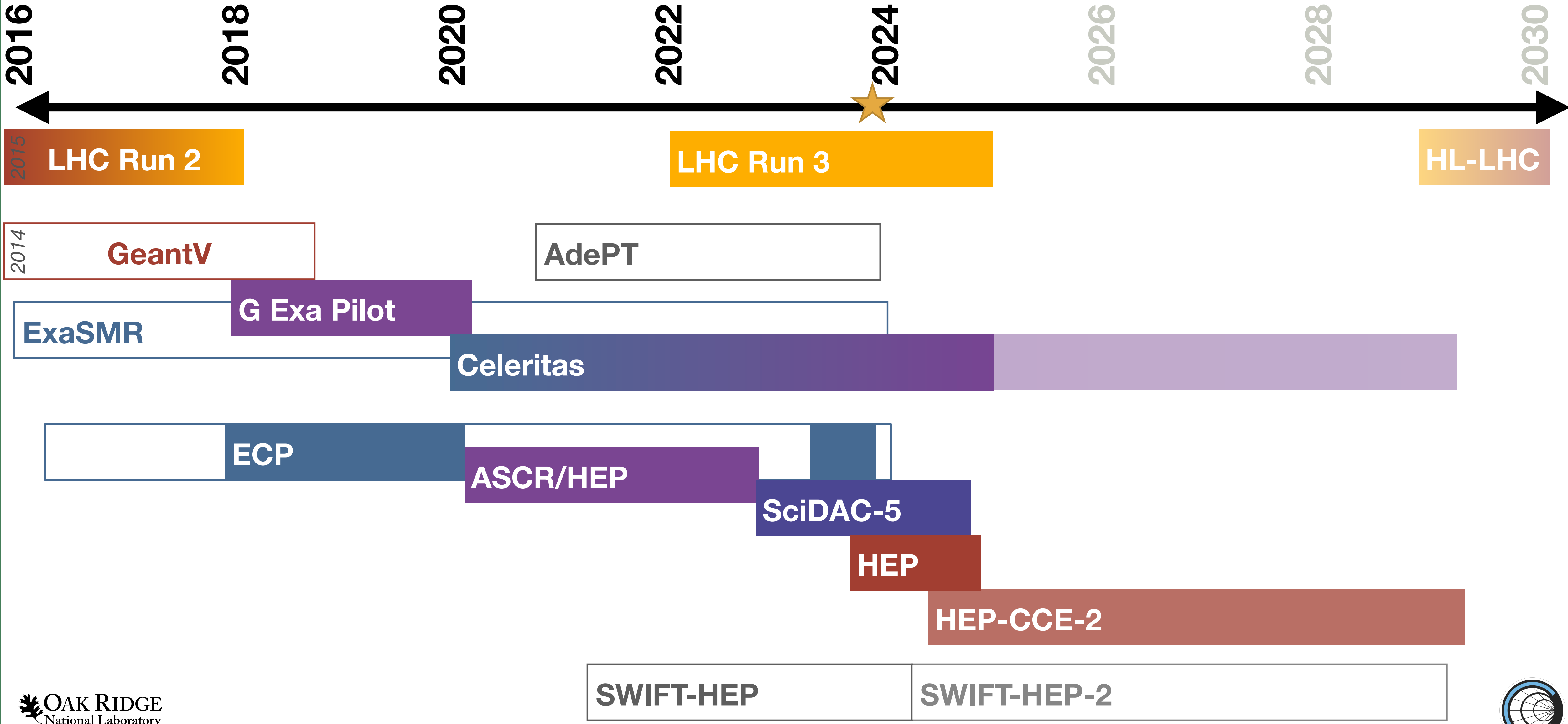
***SciDAC***: This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, Scientific Discovery through Advanced Computing (SciDAC) program.

***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.

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

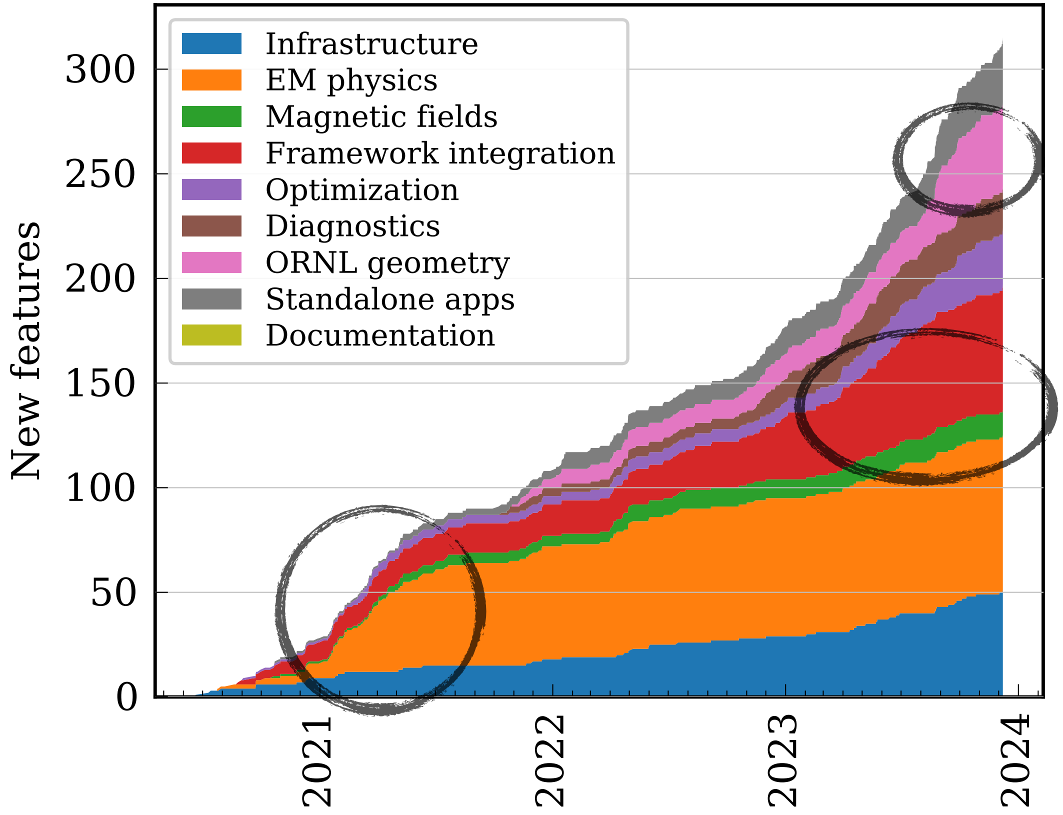
# Backup slides

# Historical context



# Code development

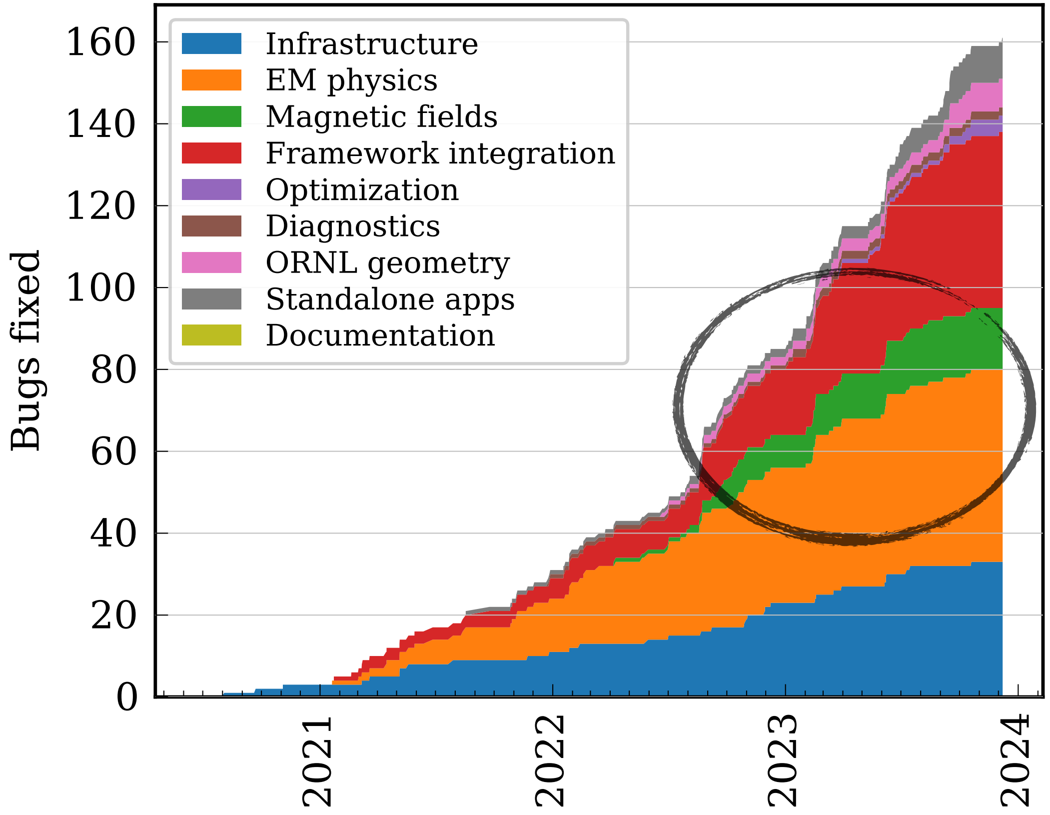
- Production-focused scientific software
  - **90%** of source code is reusable library code
  - **1:2** ratio of lines of documentation to code
  - **50k** lines of test code
- Early push for EM physics
- Last year's focus:
  - Integration with Geant4
  - Optimization on GPU (and CPU)
  - ORANGE features for ExaSMR reactor simulation




***Enhancement** pull requests*

# Code development (flip side)

- 1 fix for every 2 enhancements
- Integration campaigns critical for finding bugs/issues
  - ATLAS integration at LBL, Feb. 2023
  - CMS integration at ORNL, June 2023
- Bug fix rate is decreasing though!
  - Most fixes are for new features
  - Each PR requires a new unit test that fails without the fix and passes after



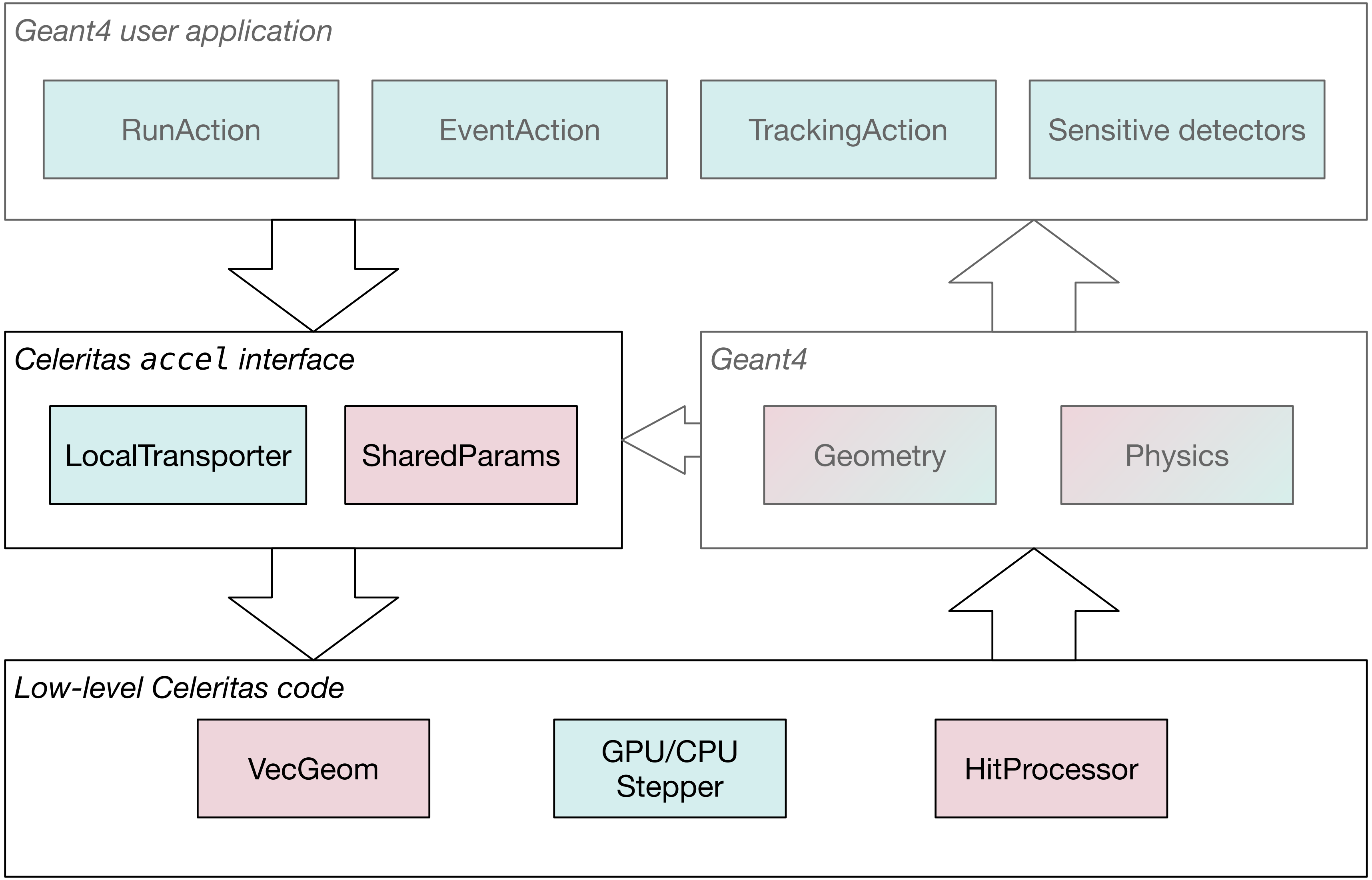

***Bug fix** pull requests*


Oak Ridge National Laboratory

# Core design philosophy

- Algorithms and structure *will* need to change due to:
  - Increasing complexity of new physics added
  - Design requirements from downstream integration
  - Performance bottlenecks found during analysis
- Therefore code needs to be amenable to refactoring
  - Heavy use of composition rather than inheritance or massive functions
  - Data-oriented to allow the same data to be reused in multiple functions
  - Template-friendly interfaces hide underlying data structures

# Geant4 interface library




*Thread-local*
*Shared*

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

# Performance per step

- Large variation in timing for early steps
  *possibly due to "looping" low-energy particles in vacuum*
- For same number of active tracks, end of simulation is 50–80% slower per step
  *likely due to geometry divergence*