Oak Ridge National Laboratory
ORNL is managed by UT–Battelle, LLC for the US Department of Energy

Celeritas: scientific software for HEP simulation

Seth R Johnson

Celeritas Code Lead
Senior R&D Staff
Scalable Engineering Applications

CELERITAS

Celeritas core team:

Elliott Biondo *(ORNL)*, Julien Esseiva *(LBNL)*, Hayden Hollenbeck *(UVA)*, Seth R Johnson *(ORNL)*, Soon Yung Jun *(FNAL)*, Guilherme Lima *(FNAL)*, Amanda Lund *(ANL)*, Ben Morgan *(U Warwick)*, Stefano Tognini *(ORNL)*

Celeritas core advisors:

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

U.S. DEPARTMENT OF ENERGY

HSF-India
1 August, 2024

Background

Methods

Results

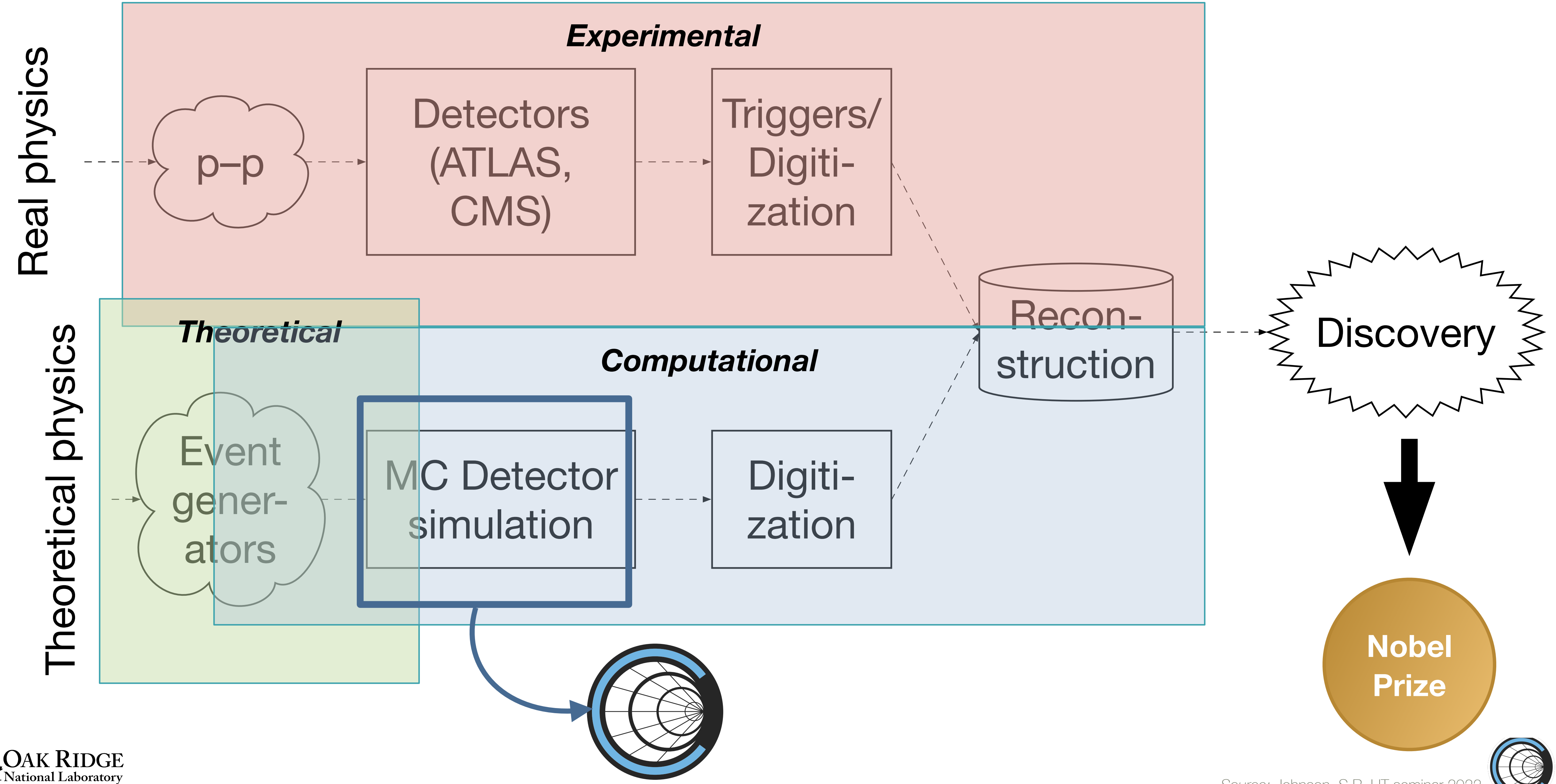
Future work

The Large Hadron Collider

- **What:** particle collider (⌀ 8.5km)
- **Where:** Geneva, Switzerland
- **Why:** study fundamental questions abo
  the nature of the universe
- **Who:** CERN and international science
  collaborations
- **How:** massive radiation detectors *(CMS
  ATLAS, …)* around collision sites

LHC Beamline ©CERN

LHC physics in a nutshell (gross simplification)

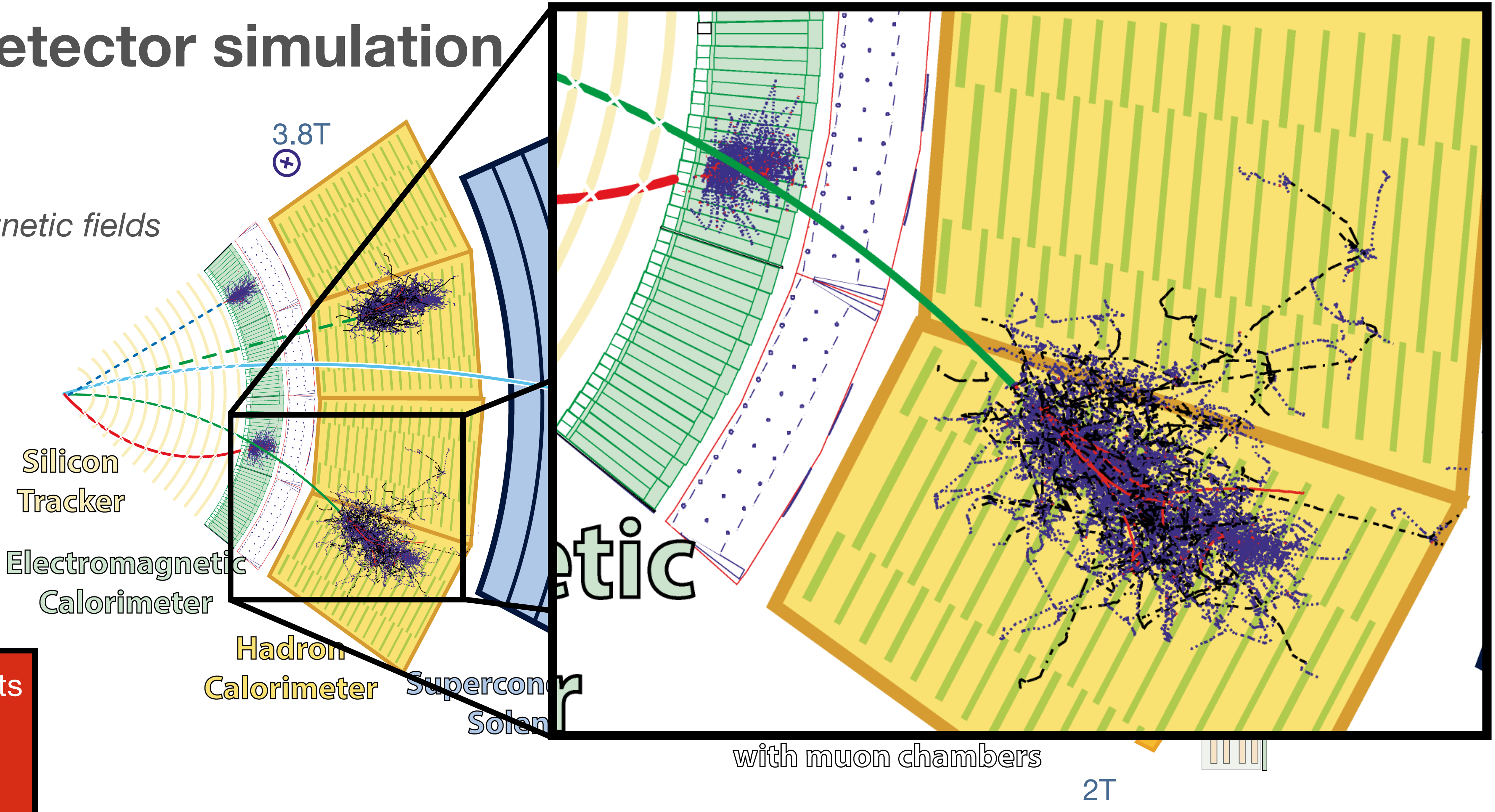


Source: Johnson, S.R. UT seminar 2022

LHC detector simulation

Note magnetic fields

1m



- **100B** events
× **1M** tracks
× **1K** steps

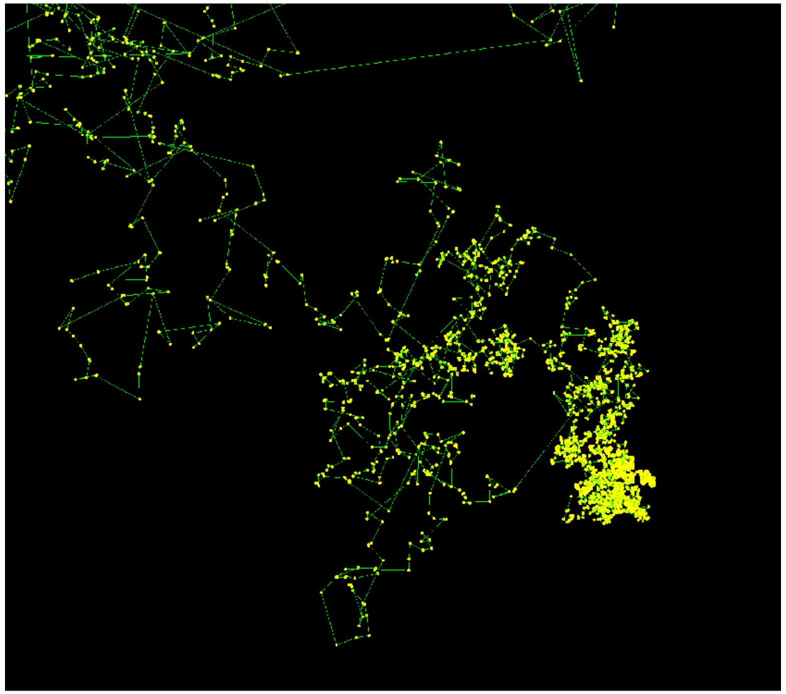
Muon **Electron** **Charged hadron (e.g. pion)**
Neutral hadron (e.g. neutron) **Photon**

https://cds.cern.ch/record/2120661

Source: Johnson, S.R. UT seminar 2022

Monte Carlo detector simulation

- Each **event** inputs a list of **primaries** (starting point for a particle track)
- Each **track** samples physical processes and may produce **secondary** particles
- Each interaction in a “sensitive detector” generates a **hit** to record output
- ~3× number of actual events (140 PB of data!) must be simulated to reduce statistical effects



In this kind of MC, each history is an analog physical realization

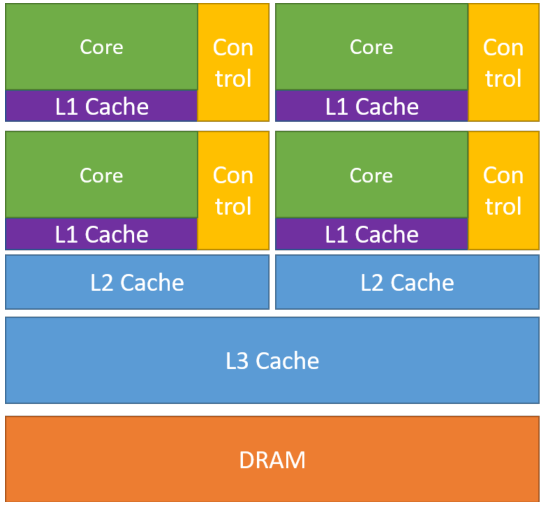
Source: Johnson, S.R. UT seminar 2022

OAK RIDGE National Laboratory

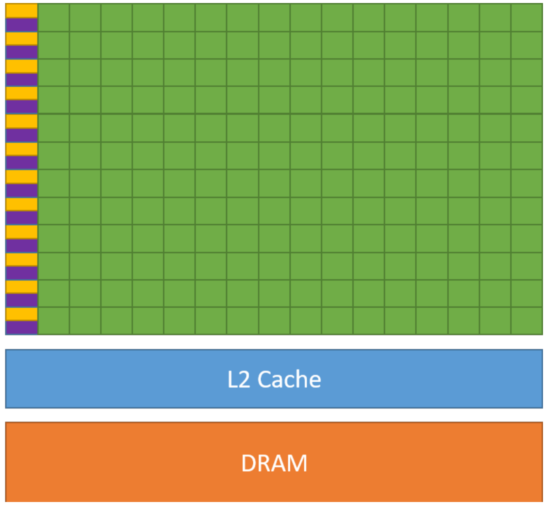
GPUs for scientific software

- 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
- Performant when:
  - Lots of similar work is being done at the same time
  - Lots of floating point operations per datum

Hardware characteristics determine programming paradigm



CPU

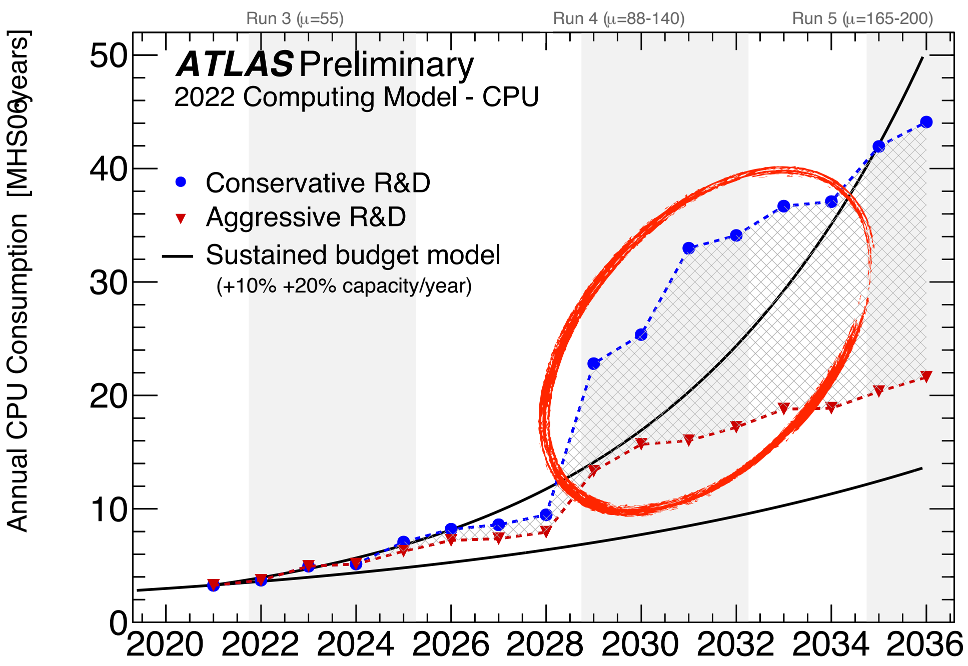


GPU

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

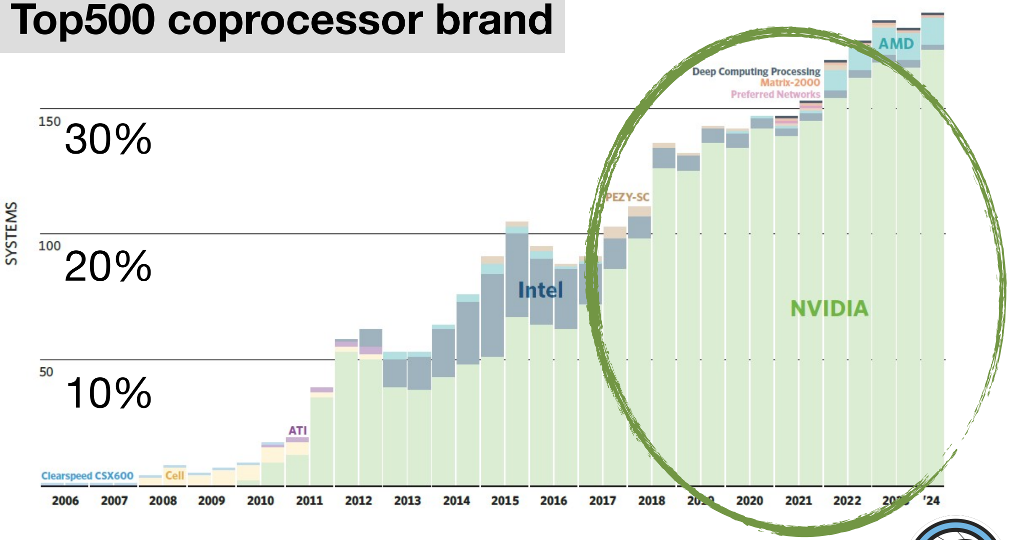
Intersection of MC + GPU

- HEP **demand** is rapidly increasing
  - Large Hadron Collider High Luminosity upgrade requires **≥10×** computational throughput for detector simulation
  - AI/ML based “fast simulation” methods need lots of training data on GPU
- HPC **supply** is fundamentally changing
  - “Heterogeneous” architectures are increasingly common in high performance computing
  - Scientific codes can run more efficiently on GPU
    *e.g., Perlmutter reports 5× average energy efficiency**
  - Demand for AI/ML training and models **will accelerate** this trend



ATLAS Collaboration: CERN-LHCC-2022-005; LHCC-G-182

Top500 coprocessor brand



Top500 Supers: This Is Peak Nvidia For Accelerated Supercomputers, Timothy Prickett Morgan, May 13, 2024.

*https://blogs.nvidia.com/blog/gpu-energy-efficiency-nersc/

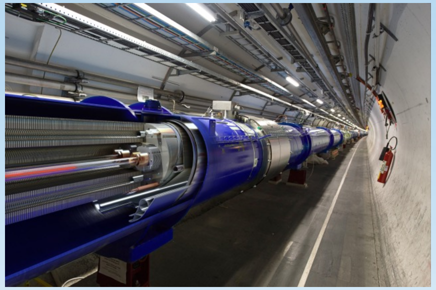
Oak Ridge National Laboratory

Goal and approach

Enable scientific discovery in HEP

by improving throughput and energy efficiency using GPU-based Monte Carlo detector simulation

- **Research and develop** novel algorithms
- **Implement** production-quality code
- **Integrate** collaboratively with experiments
- **Deploy** on DOE LCF resources



LHC beamline ©CERN

CELERITAS

Nvidia H100 GPU @Nvidia

Informal collaborators

Background

Methods

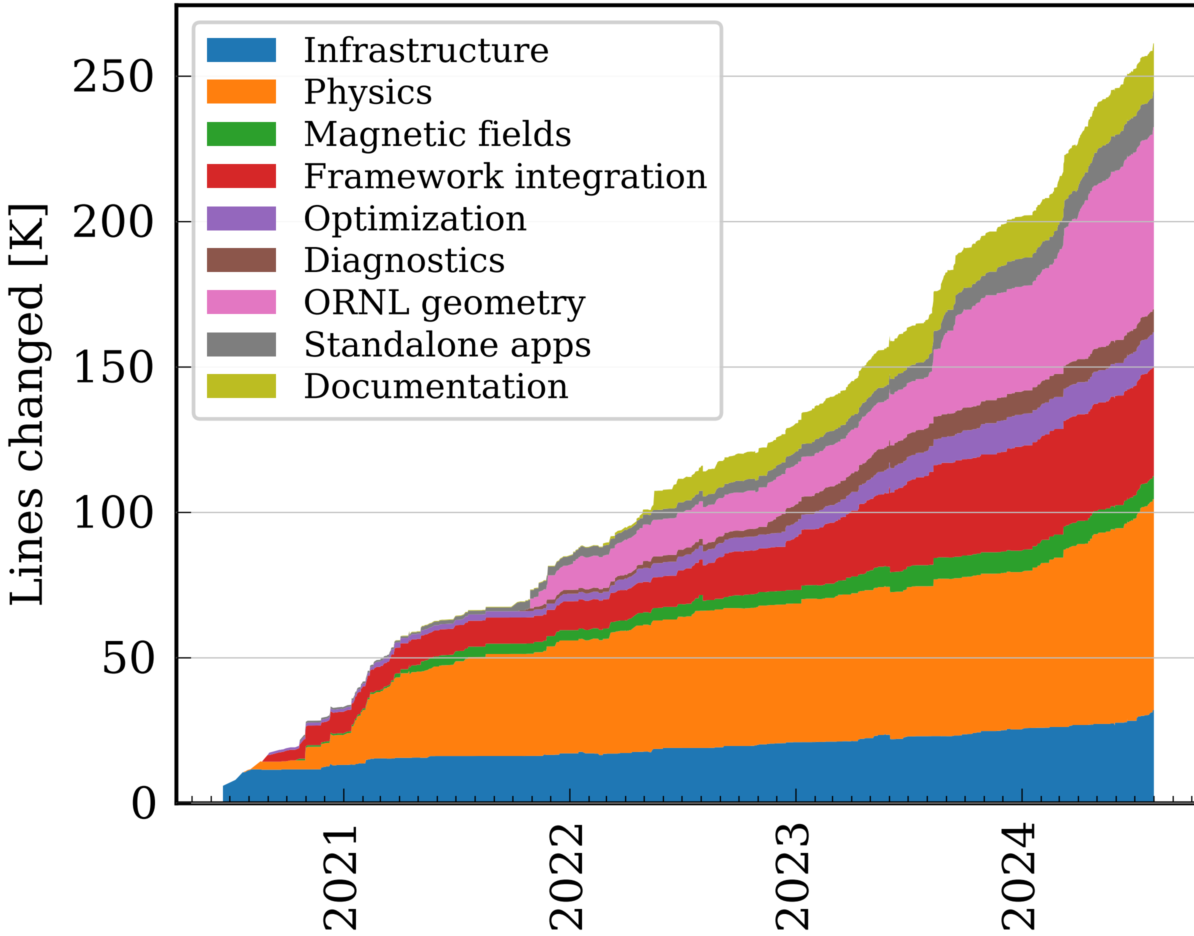
Results

Future work

Code development

Production-focused open source scientific software

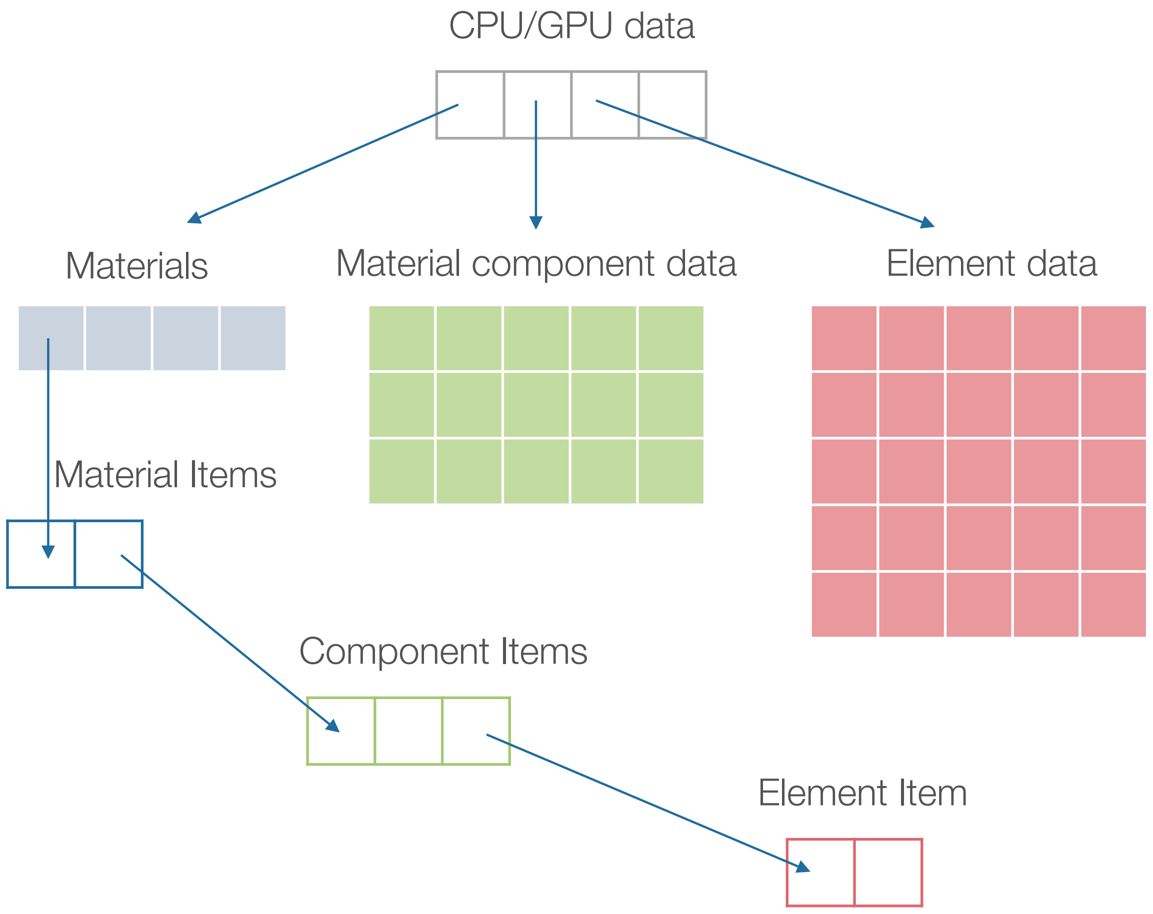
- **90%** of source code is reusable library code
- **1:2** ratio of lines of documentation to code
- **50k** lines of test code
- CI and rigorous review before merge



Oak Ridge National Laboratory

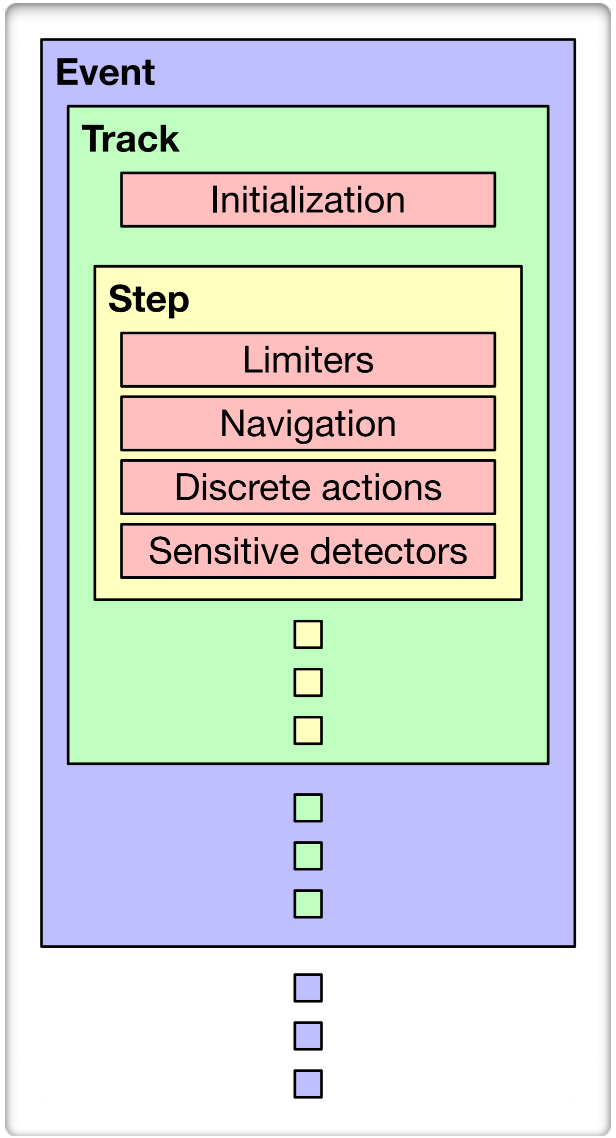
Data management

- Traditional GPU data uses:
  - Highly structured, dense, regular arrays
  - Lots of host/device transfers
- Physics data is:
  - "Awkward," hierarchical, sometimes sparse structures
  - Mostly constant after problem setup
- Celeritas data structures:
  - Run on CPU, Nvidia, AMD, …
  - Are constructed on CPU with standard C++

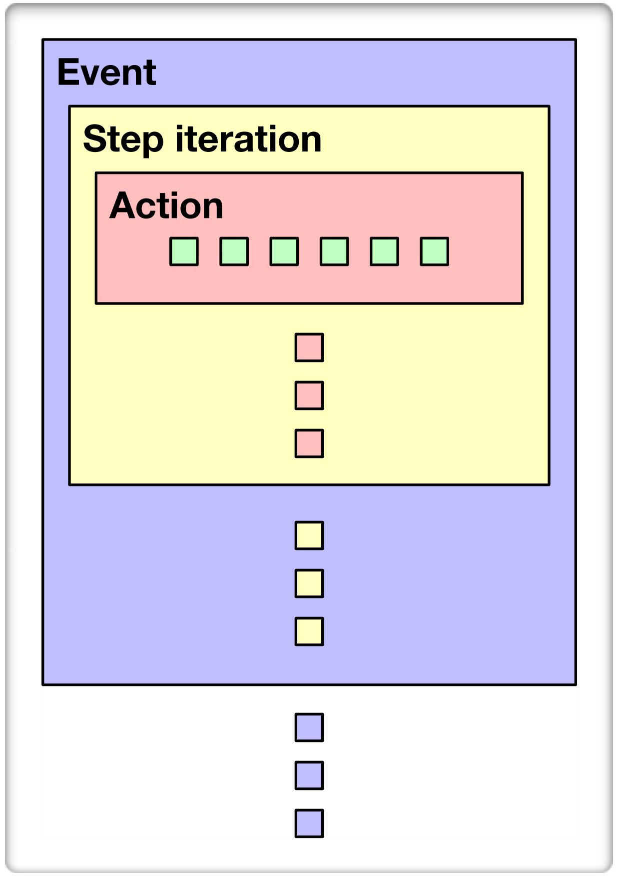


Core algorithm for simulation: stepping loop

- External synchronization point at each “event” (*p*–*p* collision)
- Dependency between steps and independence of tracks allows *loop interchange*
- Instead of polymorphic functions operating on a single track, they launch a kernel over *many* tracks



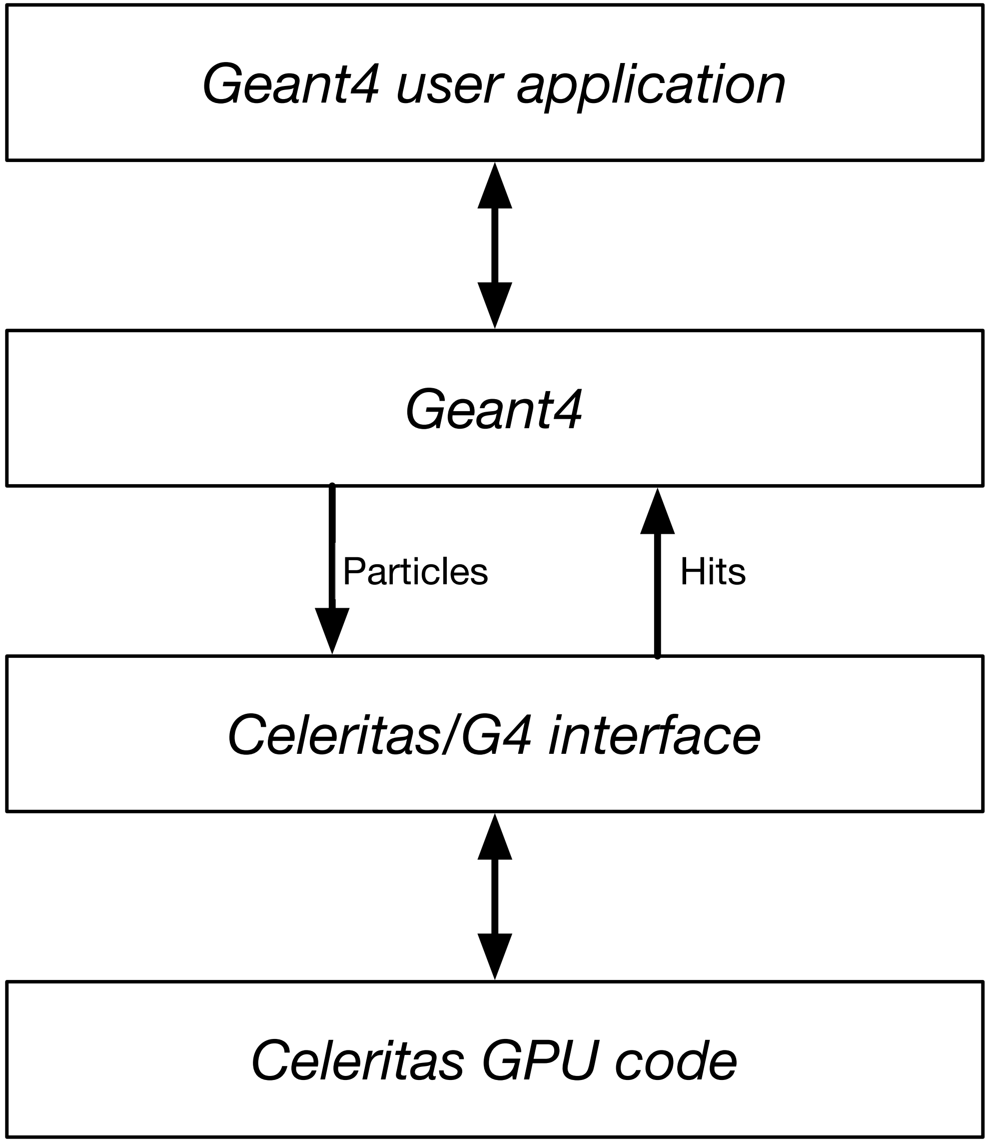
CPU (Geant4)



GPU (Celeritas)

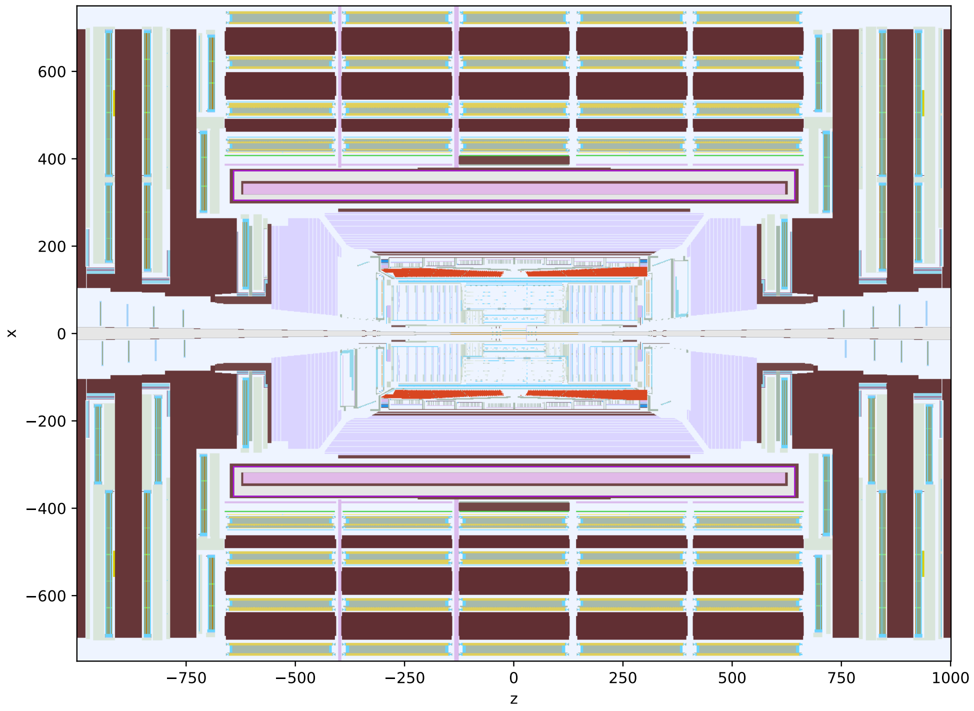
Geant4 integration

- Geant4 is the *experiment-validated* MC simulation code used in HEP and beyond *(medical physics, dosimetry, …)*
- Celeritas directly imports geometry, physics data
- e^-, e^+, γ sent to Celeritas (GPU)
- Reconstructed "hits" *(energy deposition plus metadata)* sent back to user-defined detectors



High-level capabilities targeting LHC simulation

- Equivalent to `G4EmStandardPhysics`
- 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

Background

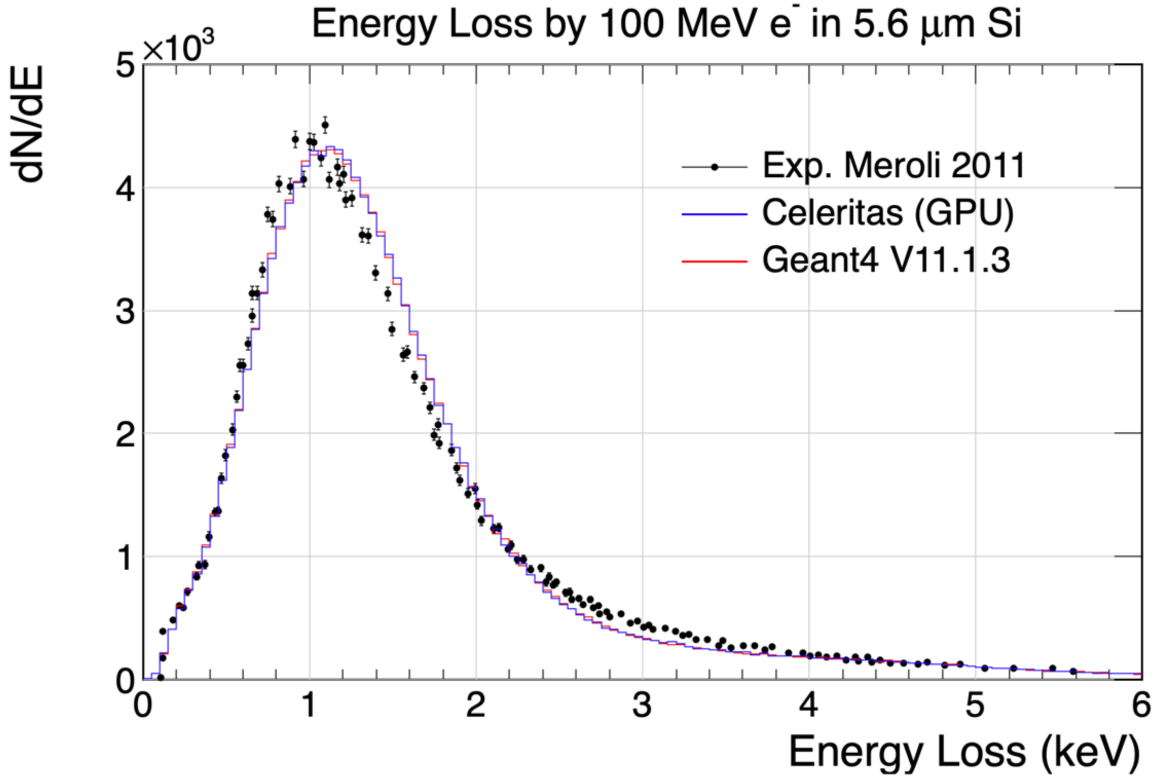
Methods

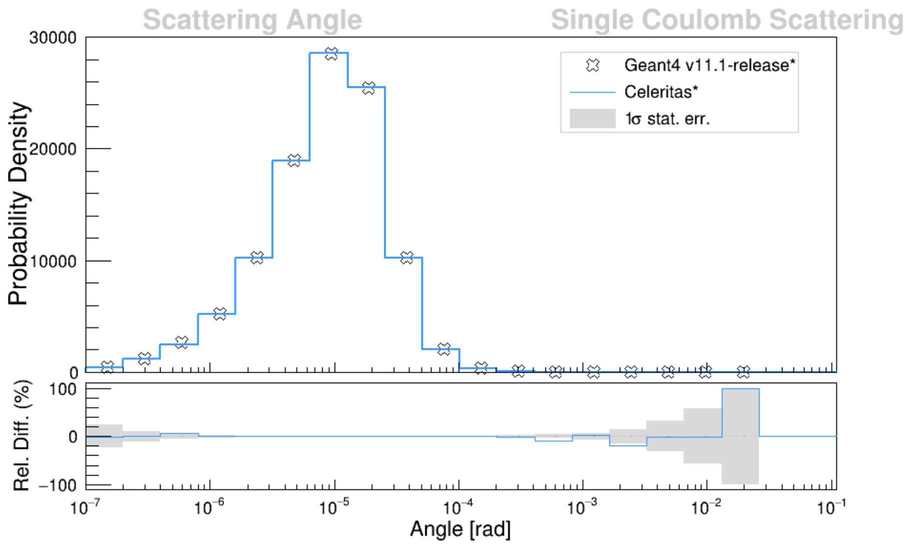
Results

Future work

EM physics validation

- Established good agreement with Geant4 for:
  - Energy loss fluctuations
  - Multiple scattering (azimuthal angle distribution)
  - Single Coulomb scattering
- Experiment-specific validation required for acceptance





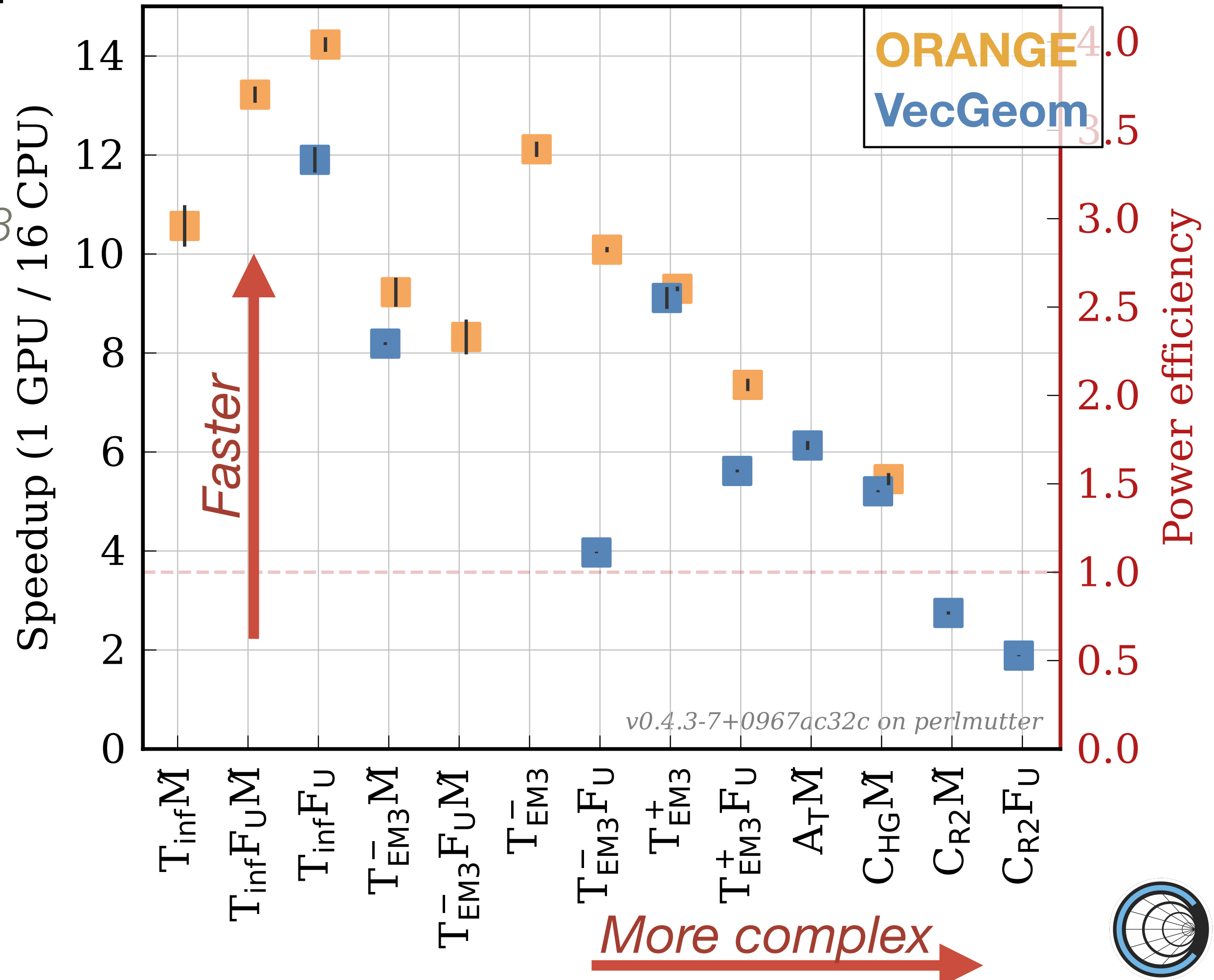
Oak Ridge National Laboratory

Standalone EM performance

Problem definition	
T_{inf}	"Infinite" medium
T_{EM3}	Idealized calorimeter
A_{TC}	ATLAS TileCal
C_{HG}	CMS HGCal
C_{R2}	CMS Run 2 (2018)

Modifier	
F_U	uniform 1T field
$\tilde{M}$	no MSC

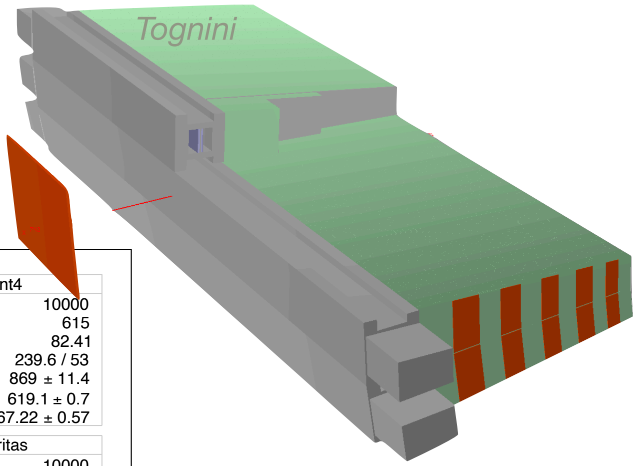
- LHC-scale simulations on DOE LCF
  - 1300 × 10 GeV e$^-$, 16 events
  - ¼ Perlmutter node (NERSC)
    1 × Nvidia A100 GPU, ¼ × 64-core AMD EPYC 7763
  - Celeritas GPU vs CPU
    CUDA (1 CPU thread) vs OpenMP (16 CPU threads)
- **Key metrics favor GPU**
  - *Capacity:* **50–94% loss** if GPUs are ignored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

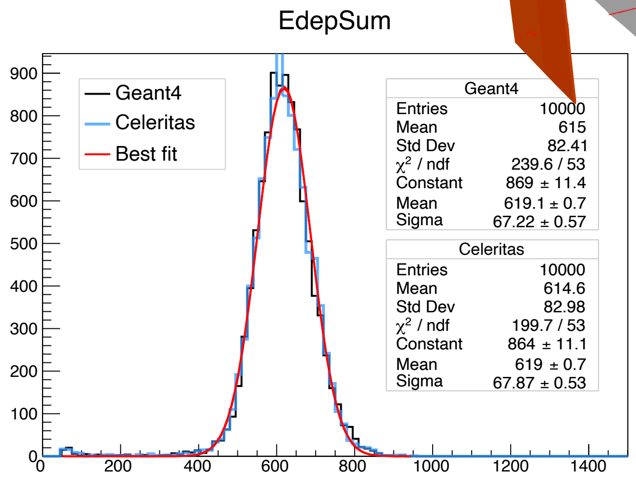


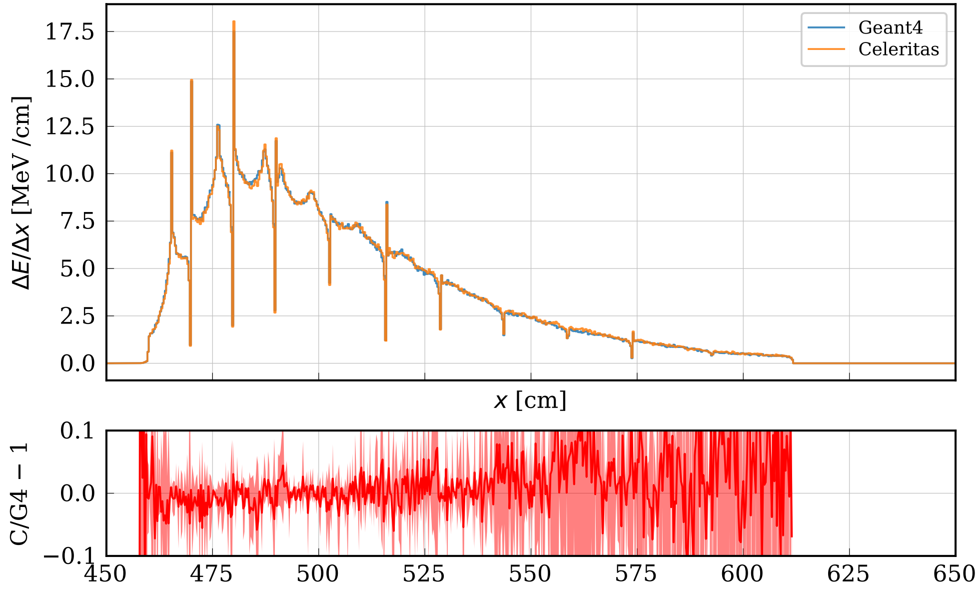
Oak Ridge National Laboratory

EM offloading with FullSimLight

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





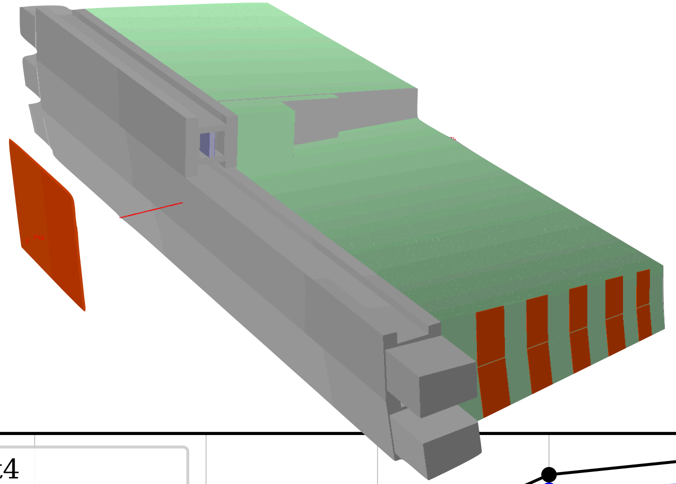


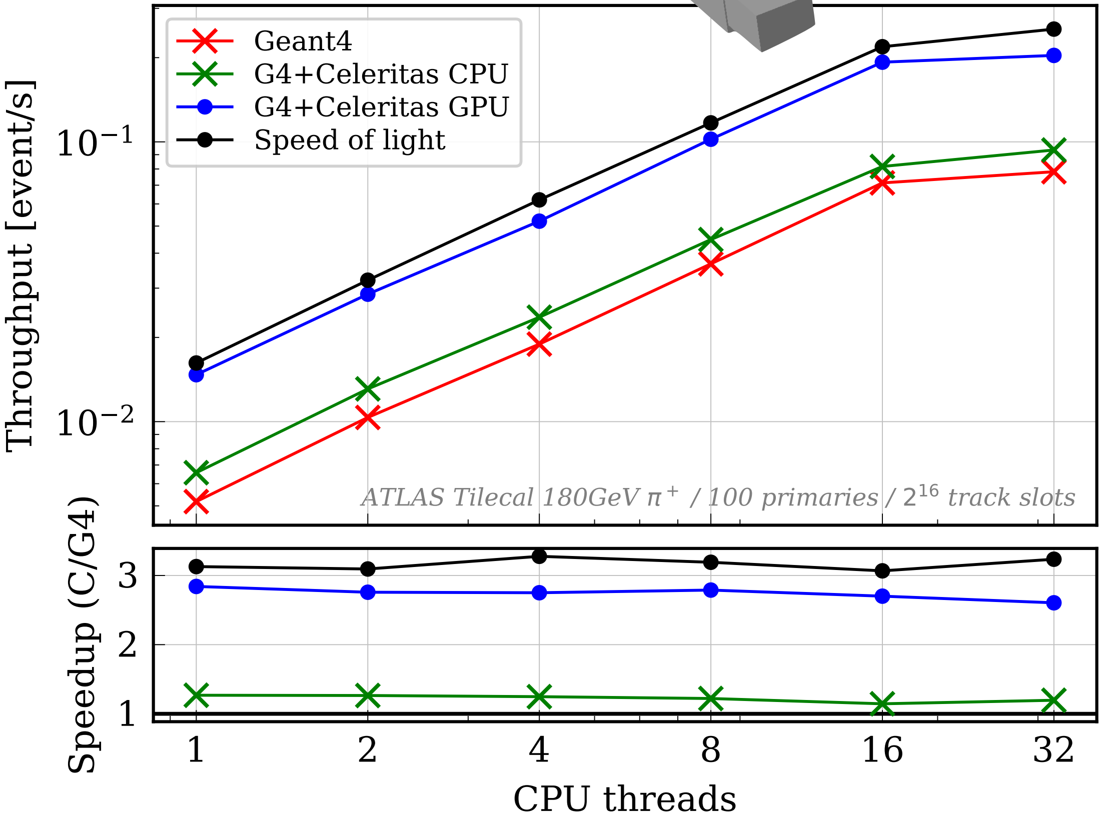
yz-integrated energy deposition

Source: Johnson, S.R. Geant4 Meeting 2023

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: **2.6–2.8×** at full occupancy
  Using all CPU cores with a single GPU
- CPU-only speedup: **1.1–1.3×**
- Theoretical maximum speedup: **3.0–3.3×**
  Instantly killing e-, e+, γ when born
- **LHC-scale energy per event (>1 TeV) is needed for GPU to be effective**
- **One GPU is effective with many-CPU Geant4**

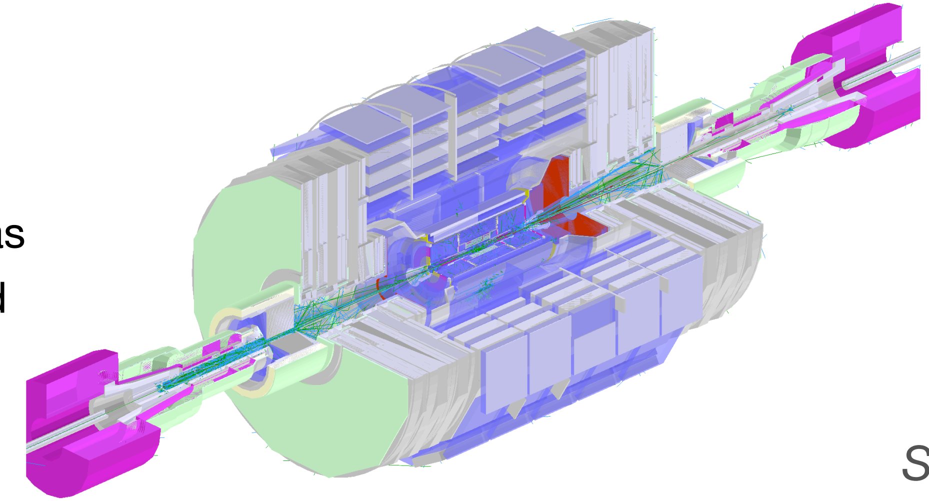




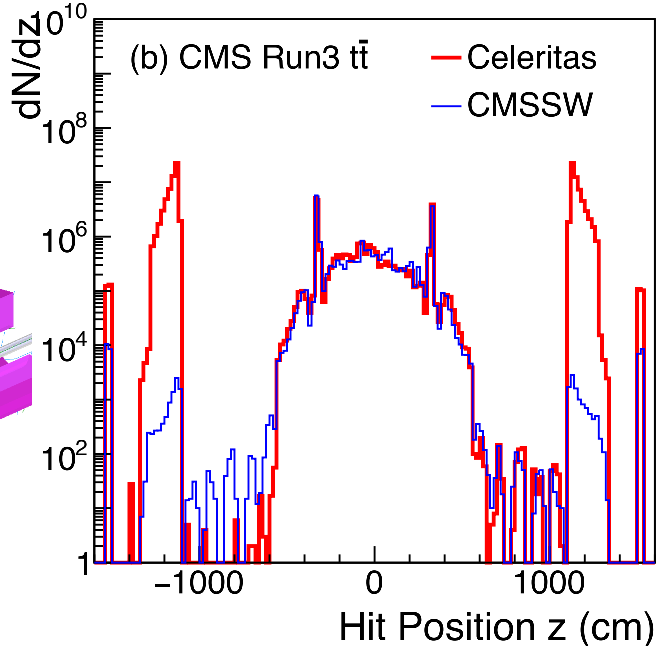
OAK RIDGE National Laboratory

CMS Run 3&4 Standalone Simulations

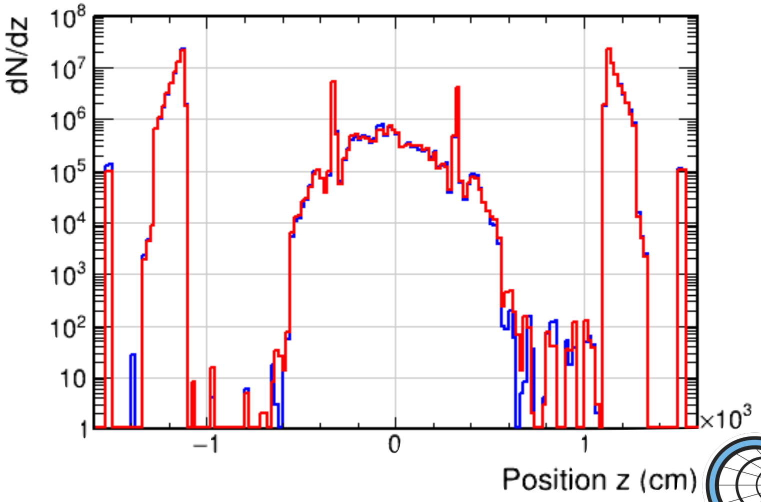
- Standalone Geant4 app `celer-g4`
- 32 $t\bar{t}$ events from Pythia
- FTFP_BERT physics
  - Geant4 simulates hadronics
  - All EM tracks offloaded to Celeritas
  - Lepto-nuclear reactions neglected
- Multiple field options
  - No magnetic field
  - Uniform 4T field
  - Discretized+interpolated RZ field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



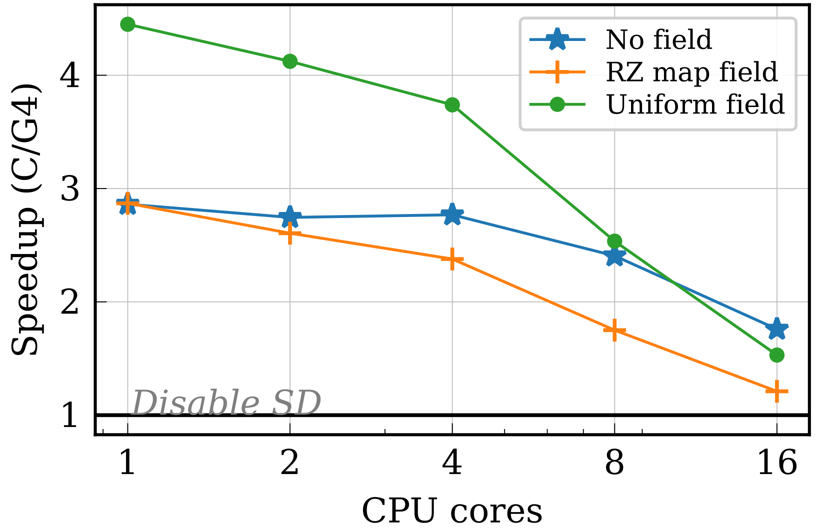
Standalone Run 3 hit distribution



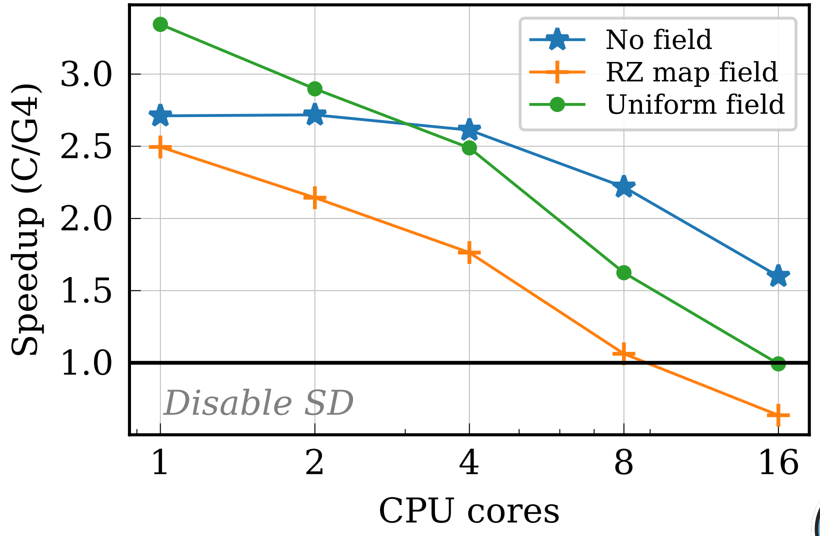
CMS Run 3&4 Standalone Results

- Promising performance
  - SD reconstruction adds <15% overhead
  - Initial comparison of hits shows good agreement
  - 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)

Run 3; Nvidia A100



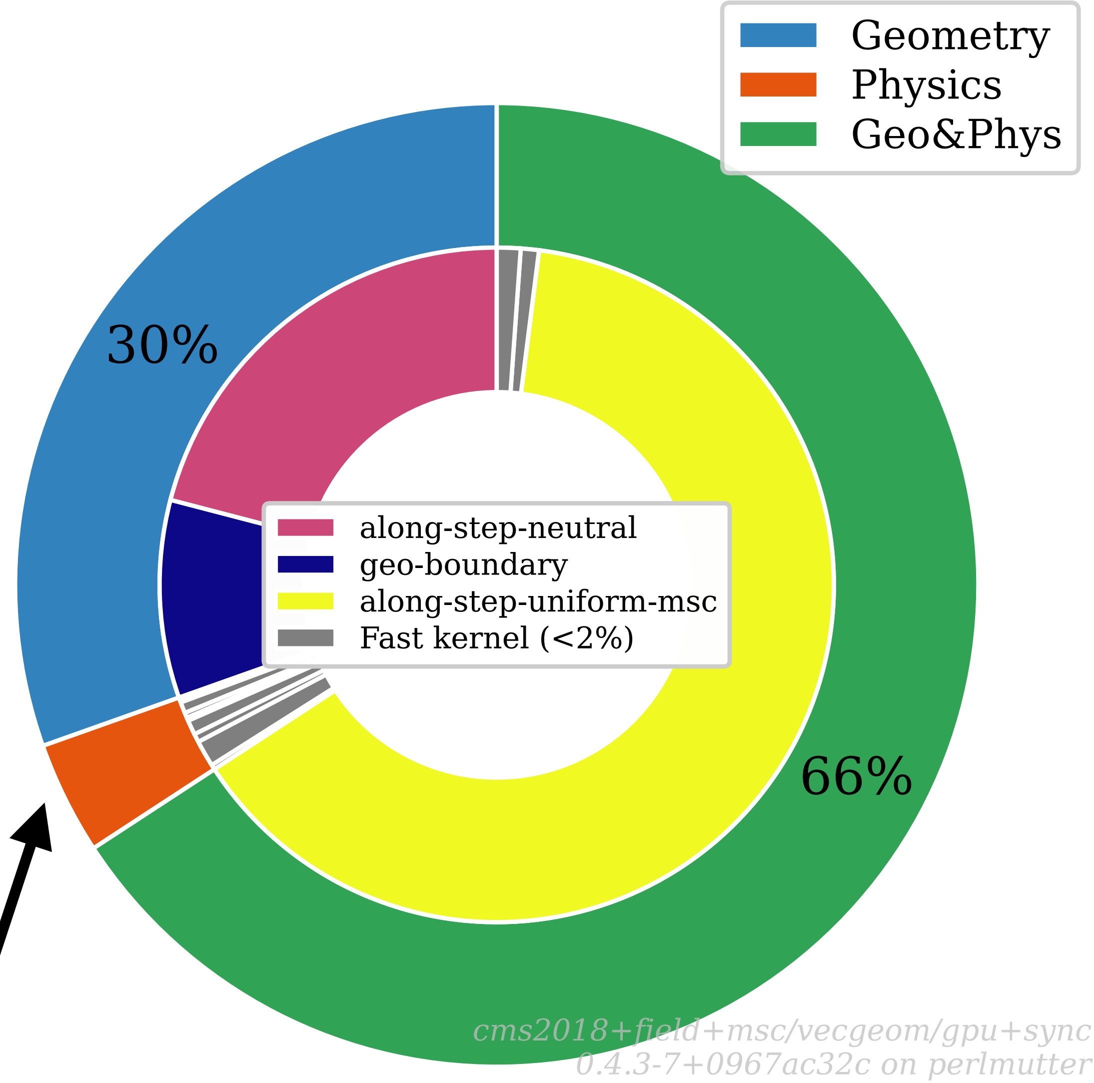
Run 4 (HL-LHC); Nvidia A100



Nvidia A100 vs AMD 7532 EPYC

Primary bottleneck: geometry

- Each step* may require 100 "distance to boundary" evaluation
  * remember, ~1B steps per simulation!
- Up to ~10^5 distinct geometric elements per detector model
- Current geometry implementation is *not* optimized for GPU
- CERN (VecGeom) and ORNL (ORANGE via HEP-CCE2) both implementing solutions



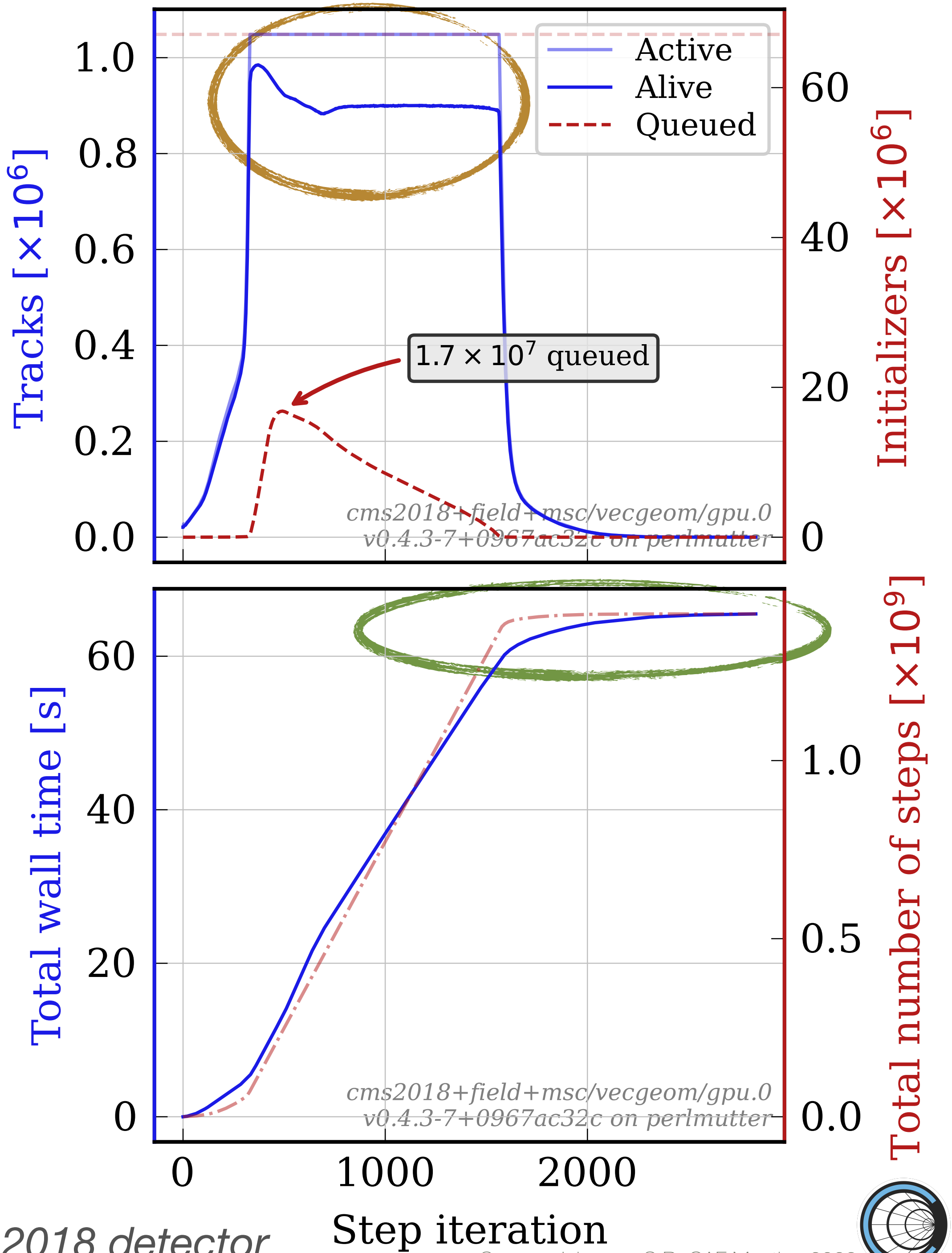
Physics is 4% on GPU, but 19% on CPU

Oak Ridge National Laboratory

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

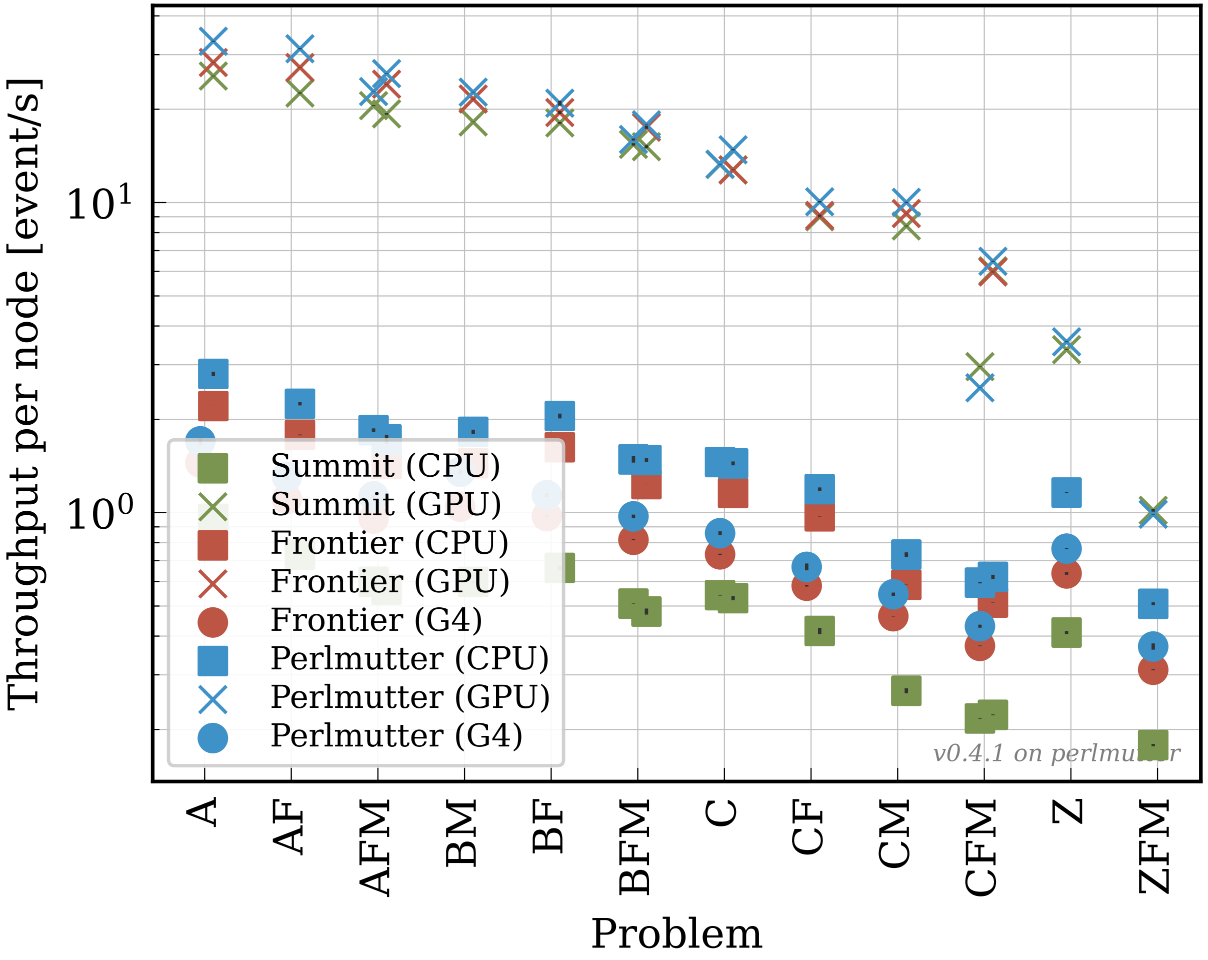
Source: Johnson, S.R. CAF Meeting 2023

Figure of merit: throughput

- GPUs *cannot* be ignored if present
- AI/ML "revolution" guarantees more coprocessors at all scales

Per-node stats for DOE supercomputers

Machine	Arch	Card	TDP (W)	Cores*	Cards
Summit	CPU	IBM Power9	190	‡22	2
	GPU	Nvidia V100	250	80	6
Perlmutter	CPU	AMD EPYC 7763	280	64	1
	GPU	Nvidia A100	250	108	4
Frontier	CPU	AMD EPYC 7453	225	‡64	1
	GPU	AMD MI250x	500	220	†4



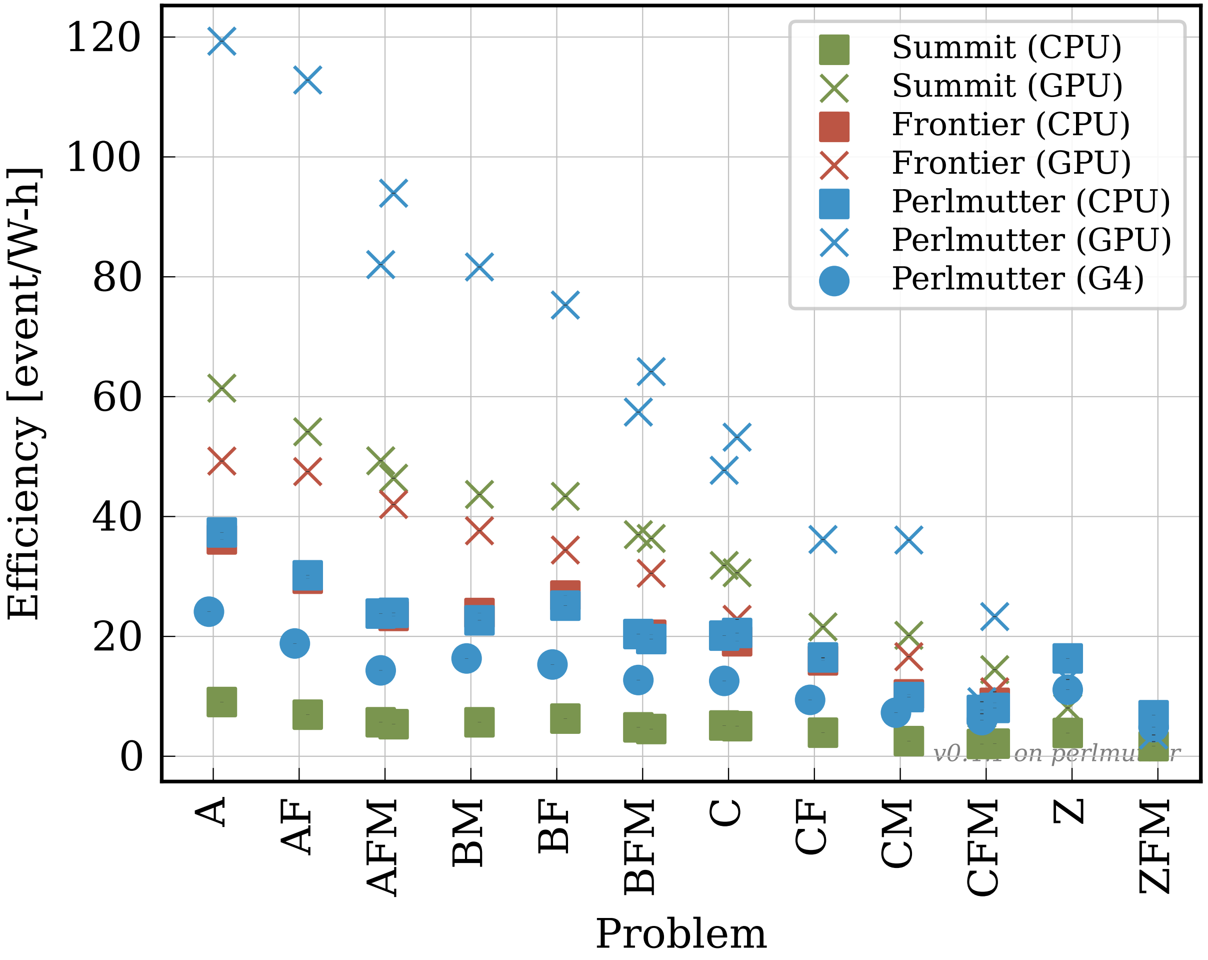
EM only, no SDs

**or SMs;*
†Each card has 2 GPUs
‡One core reserved per GPU

Oak Ridge National Laboratory

Figure of merit: efficiency

- Estimated using reported Thermal Design Power (TDP)* and Celeritas throughput
- GPU consistently shows higher energy efficiency 🌱
  - Reduced operating costs
  - Higher compute density (fewer nodes, smaller data centers)
- A100:EPYC price is ~4× 💸



*May be conservative based on nvidia-smi readings

EM only, no SDs

Results: impact by the numbers

100	**lines of code**	to integrate Celeritas into a FullSimLight tile calorimeter test application, with no modifications to Geant4
2.8×	**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/

Background
Methods
Results
Future work

Python/REST interface

Challenge: generic integration with next-generation experiment frameworks

- JSON I/O in C++
  - Integrates cleanly with containers
  - Quickly “spin up” interactive notebooks for student exploration
- Current capability: rasterization for geometry validation

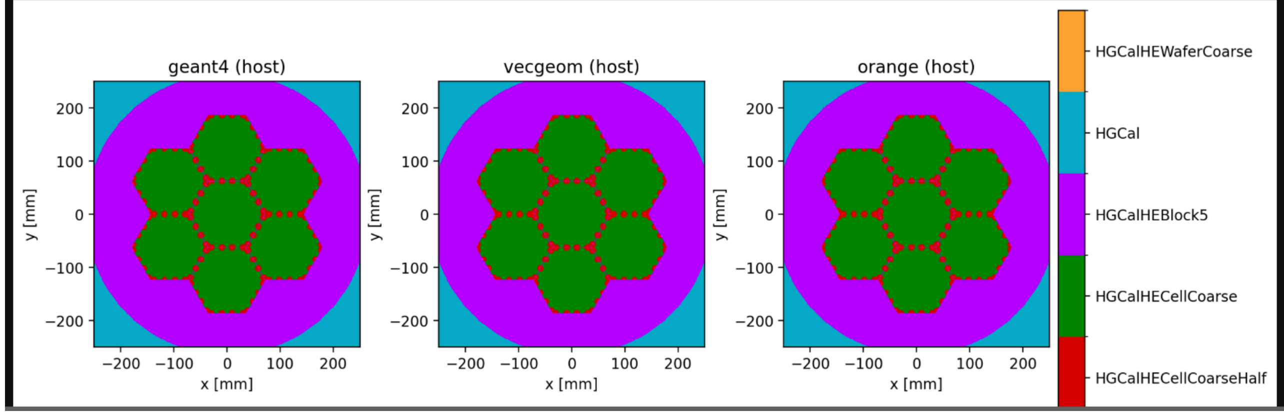
```python
from itertools import count
from celerpy import model, visualize

plot_all_geometry = visualize.plot_all_geometry
```

```python
celer_geo = visualize.CelerGeo.from_filename("../../models/TBHGCal181Oct.gdml")
```

```
info: Reading JSON line input from <stdin>
info: Loading Geant4 geometry from GDML at ../../models/TBHGCal181Oct.gdml
```

```python
im = plot_all_geometry(visualize.Imager(
    celer_geo,
    model.ImageInput(
        lower_left=[-25, -25, 2602.06],
        upper_right=[25, 25, 2602.06],
        rightward=[1.0, 0.0, 0.0],
        vertical_pixels=527,
    )),
figsize=(12,4))
```

```
status: Tracing geant4 image on host
info: Writing image to '/var/folders/n9/mqnx20b929z469f6p3fbq7c40000gn/T/tmpuuab_rgj.bin'
status: Tracing vecgeom image on host
info: Writing image to '/var/folders/n9/mqnx20b929z469f6p3fbq7c40000gn/T/tmpf57l3gdy.bin'
status: Tracing orange image on host
info: Writing image to '/var/folders/n9/mqnx20b929z469f6p3fbq7c40000gn/T/tmpkk4wht5l.bin'
```



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

Platform-portable surface-based geometry

Challenge: most compute intensive aspect of EM simulation on GPU

✓Model conversion

✓Robust surface construction

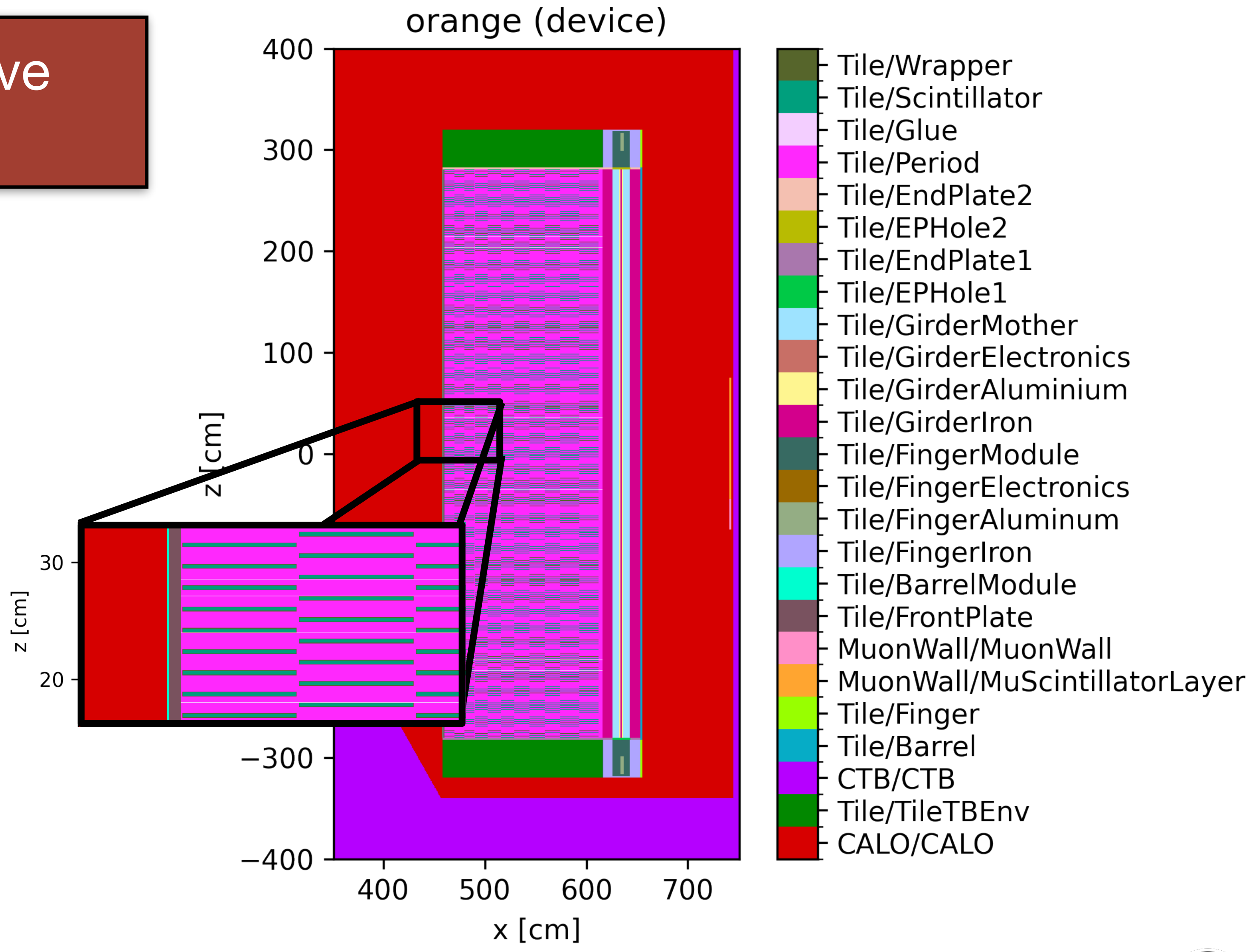
➡Model verification

➡Performance optimization

× Safety calculation

Initial targeted geometry

- ATLAS TileCal
- CMS HGCal
- ATLAS EMEC



GPU ORANGE raytrace of ATLAS tile calorimeter

Optical photon transport for Calvision

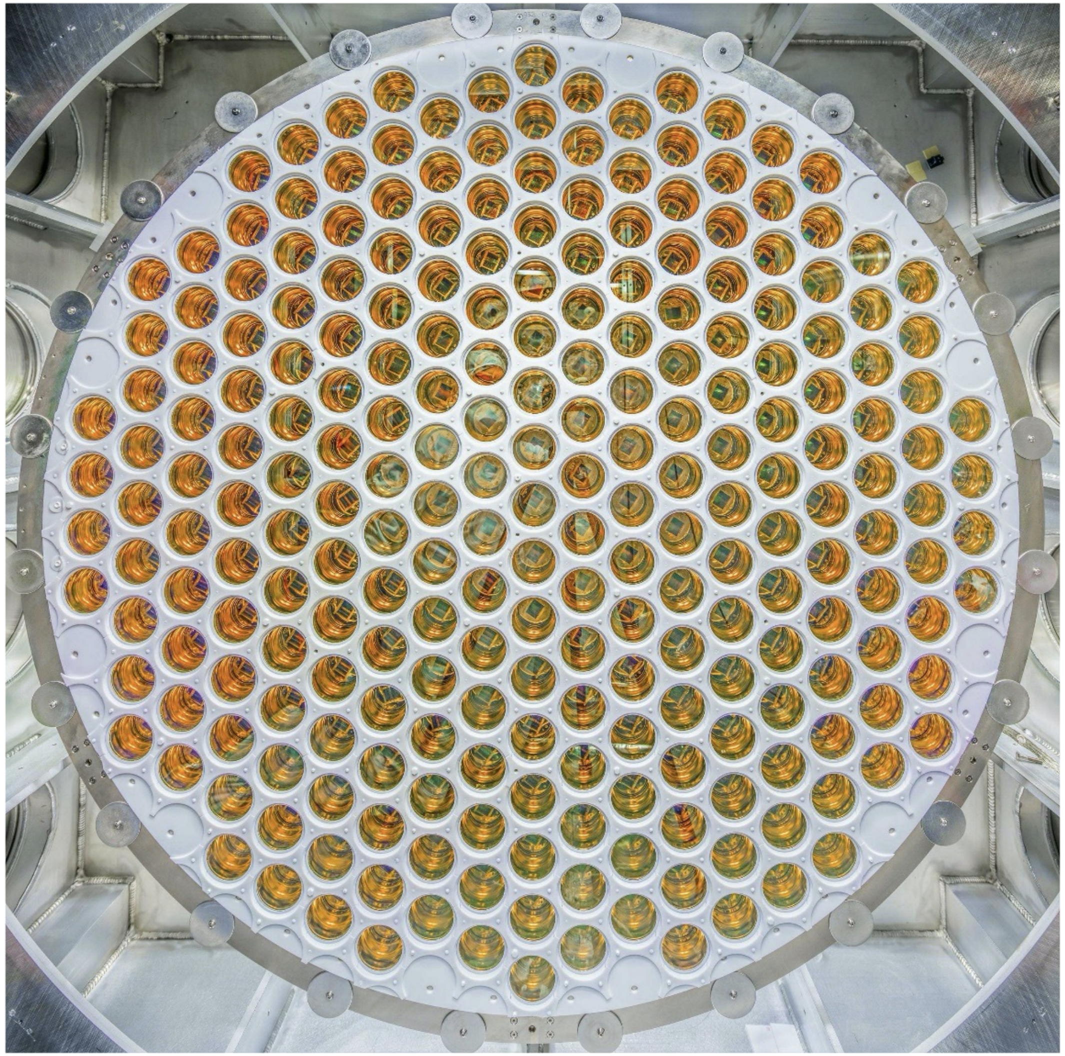
Challenge: thousands of optical photons can be emitted per track per step, leading to long run times

- **Initial goal:** integrated optical tracking loop with absorption by end of summer

✓Geant4 optical data import

✓Scintillation production

✓Cerenkov production



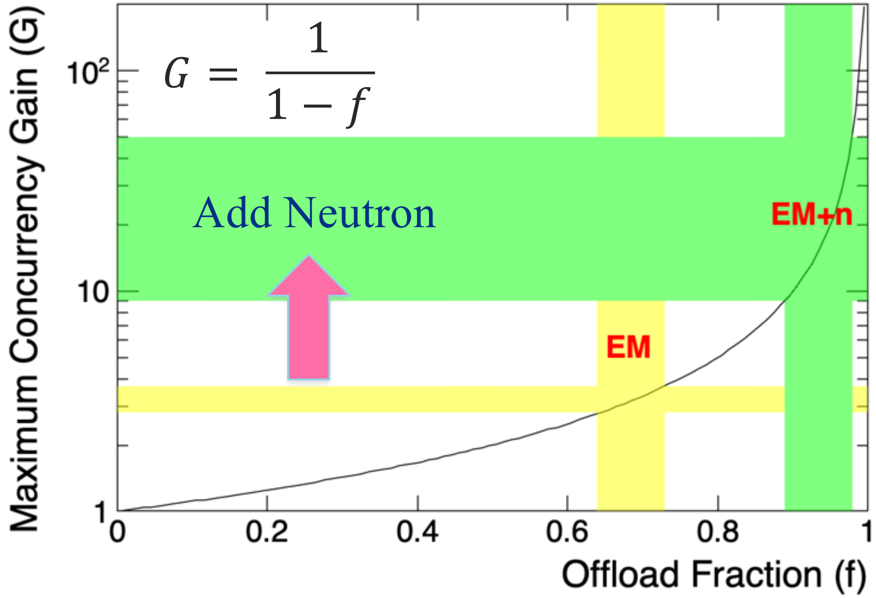
PMT array ©LZ

Neutron physics

Challenge: offloading more work to GPU

- Critical for muon beam background simulations
  (AI/ML training: see K. Pedro ECA proposal)
- **Technically feasible** due to early investment and design decisions

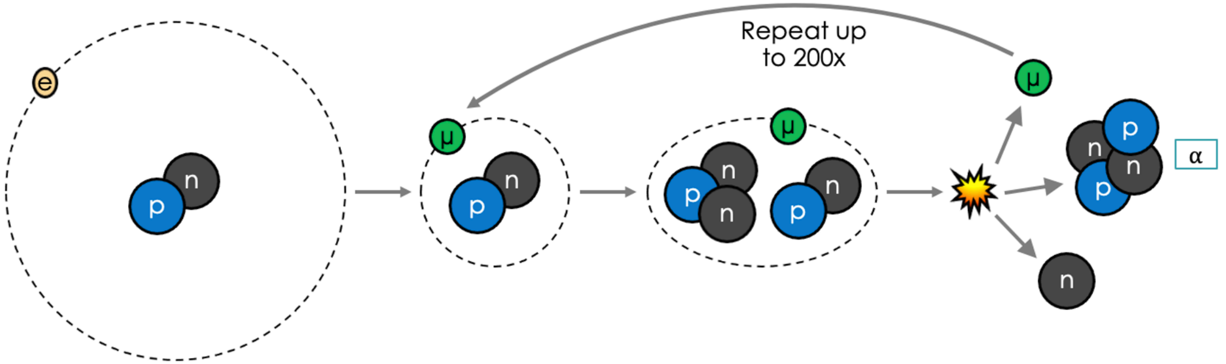
cms2018@LHC	Max Gain by GPU	
Process	EM	EM+n
QCD(p_T^{min}=1TeV)	2.9	21.1
W+Jets	2.8	22.2
Z+Jets	3.0	22.7
$t\bar{t}$	2.8	21.0
Higgs	2.8	21.1
MSSM	3.4	20.2
Stop	2.8	20.9



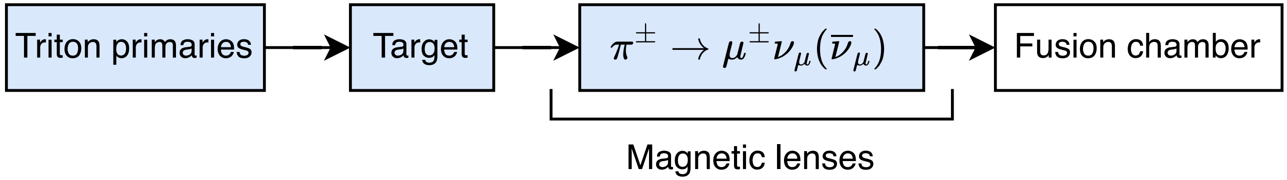
S Yung Jun, HSF Detector Simulation on GPU, May 2022

Physics for muon-confined fusion

Challenge: improving beam simulation throughput for design optimization



- Design optimization and to validation of theoretical models against experimental data
- R&D to address the feasibility of μCF as an energy source



μCF takes advantage of the reduced size of d-t muonic molecules to achieve fusion at low temperature regimes

Source: Tognini, S

Continuing collaborations

- Experiments for integrating
- Institutions for strategizing
- Codes for knowledge sharing
- Vendors for performance benchmarking

Any and all contributors and collaborators are welcome!

Interested?

Code

- Check out our GitHub repository
  - ★ to show interest and get updates
  - Easy installation and thorough documentation
  - Code structure is conducive to student projects
- Standalone “starter” tasks available
  - Physics verification and code-to-code comparisons
  - Geometry development
  - Physics model implementations

Documentation

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)
- 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.

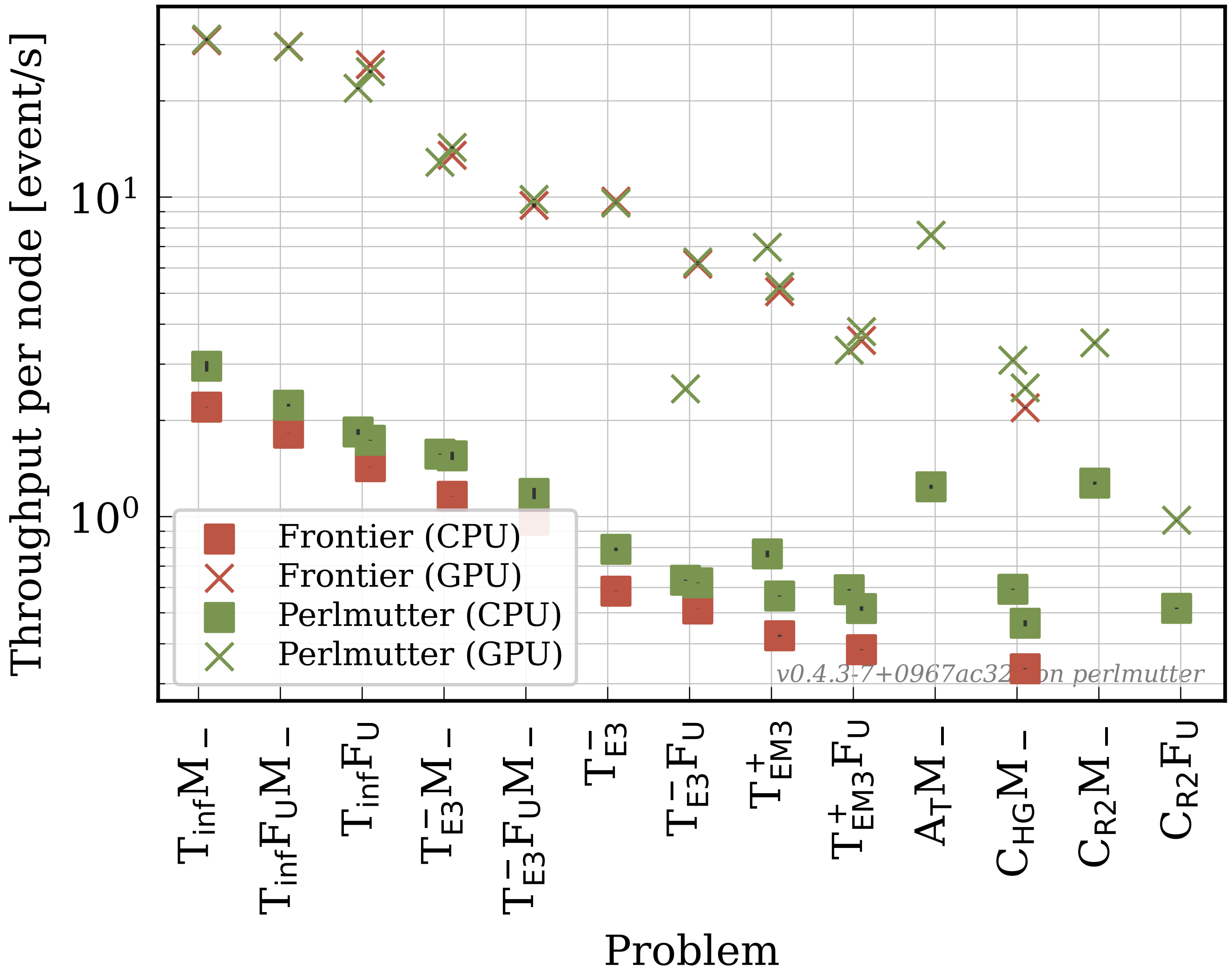
SciDAC: This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, Scientific Discovery through Advanced Computing (SciDAC) program.

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 | *Documentation* | *Publication*

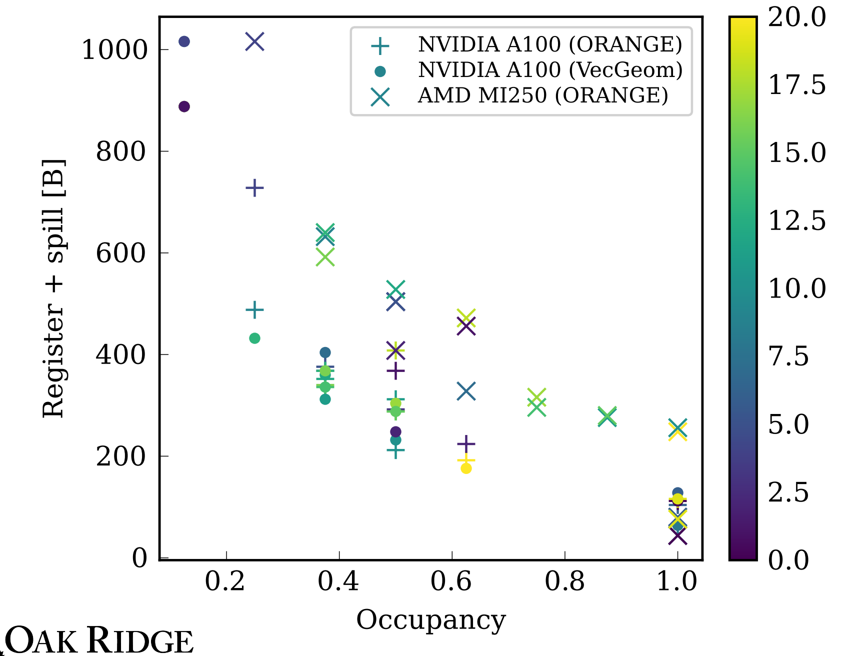
BACKUP SLIDES

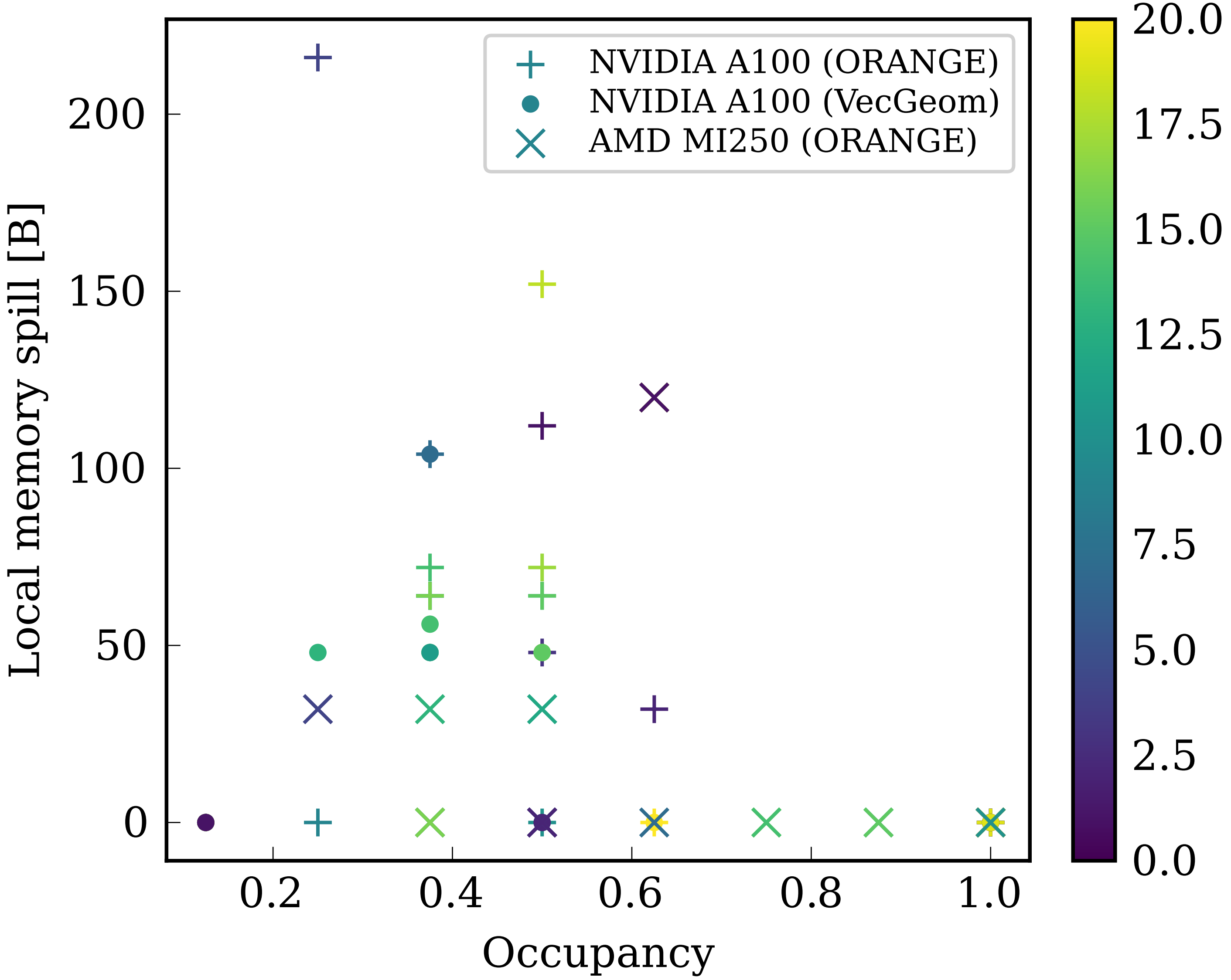
Computational throughput



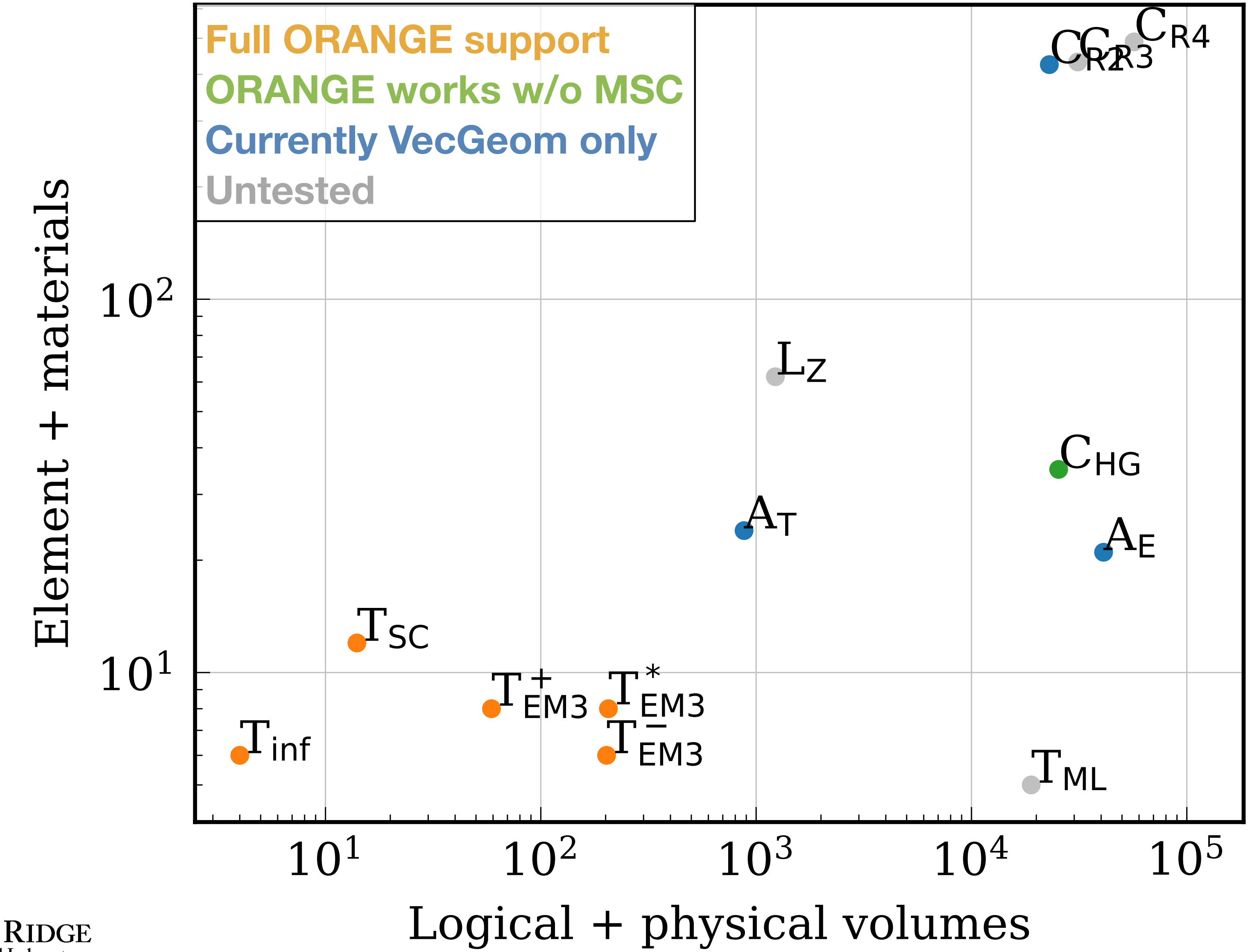
GPU Kernel occupancy

- Geometry is register-intensive
- VecGeom requires additional dynamic stack (up to 32KB/thread!)





Test problem complexity



name	label
Infinite	T_{inf}
TestEM3 (flat)	T^-_{EM3}
TestEm3 (composite)	T^+_{EM3}
TestEm3 (expanded)	T^*_{EM3}
Simple CMS	T_{SC}
ATLAS EMEC	A_E
ATLAS TileCal	A_T
CMS HGCal	C_{HG}
CMS Run 2 (2018)	C_{R2}
CMS Run 3 (2022)	C_{R3}
CMS Run 4 (HL)	C_{R4}
TrackML	T_{ML}
ALICE ZDC	L_Z