Celeritas: GPU-accelerated particle transport for detector simulation in High Energy Physics experiments *

S. C. Tognini† [1], P. Canal[2], T. M. Evans[1], G. Lima[2], A. L. Lund[3], S. R. Johnson[1], S. Y. Jun[2], V. R. Pascuzzi[4], and P. K. Romano[3]

[1]Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[2]Fermi National Accelerator Laboratory, Batavia, IL 60510, USA
[3]Argonne National Laboratory, Lemont, IL 60439, USA
[4]Brookhaven National Laboratory, Upton, NY 11973, USA

ABSTRACT

Within the next decade, experimental High Energy Physics (HEP) will enter a new era of scientific discovery through a set of targeted programs recommended by the Particle Physics Project Prioritization Panel (P5), including the upcoming High Luminosity Large Hadron Collider (LHC) HL-LHC upgrade and the Deep Underground Neutrino Experiment (DUNE). These efforts in the Energy and Intensity Frontiers will require an unprecedented amount of computational capacity on many fronts including Monte Carlo (MC) detector simulation. In order to alleviate this impending computational bottleneck, the *Celeritas* MC particle transport code is designed to leverage the new generation of heterogeneous computer architectures, including the exascale computing power of U.S. Department of Energy (DOE) Leadership Computing Facilities (LCFs), to model targeted HEP detector problems at the full fidelity of Geant4. This paper presents the planned roadmap for *Celeritas*, including its proposed code architecture, physics capabilities, and strategies for integrating it with existing and future experimental HEP computing workflows.

Submitted to the Proceedings of the US Community Study
on the Future of Particle Physics (Snowmass 2021)

†Corresponding author, togninis@ornl.gov

1 Introduction

HEP is entering an exciting era for potential scientific discovery. A targeted program, as recommended by the P5 report [1], addresses the main science drivers in all three DOE HEP Frontiers: Energy, Intensity, and Cosmic. Two of these flagship projects are the upcoming high luminosity upgrade of the Large Hadron Collider (HL-LHC) and its four main detectors, and DUNE at the Sanford Underground Research Facility (SURF) and Fermi National Accelerator Laboratory (Fermilab). These detectors will achieve a much higher granularity while also being exposed to higher readout rates than previous experiments, leading to an increase in data volume and complexity by orders of magnitude. As a consequence, there are challenges in both online and offline computing infrastructures that need to be overcome in order to store, process, and analyze this data.

Precision measurements and new physics discoveries rely heavily on comparisons between recorded data and highly detailed MC simulations. The required quantity of simulated MC data can be ten times greater than the experimental data to reduce the influence of statistical effects and to study the detector response over a very large phase space of new phenomena. Additionally, the increased complexity, granularity, and readout rate of the detectors require the most accurate—and thus most compute intensive—physics models available. Therefore, these new facilities will require a commensurate increase in computational capacity for the MC detector simulations necessary to extract new physics. However, projections of the computing capacity available in the coming decade fall far short of the estimated capacity needed to fully analyze the data from the HL-LHC [2, 3]. The contribution to this estimate from MC full detector simulation is based on the performance of the current state-of-the-art and LHC baseline MC application Geant4 [4–6], a threaded CPU-only code whose performance has stagnated with the deceleration of clock rates and core counts in conventional processors. Overcoming this bottleneck by improving the performance of Geant4 or replacing components of its simulation with lower-fidelity "fast" models, such as FastCaloSim [7], or artificial intelligence (AI)-based methods such as FastCaloGAN [8], is seen as a critical pathway to fulfilling the computational requirements of the HL-LHC [9].

Instead of relying on fast models to replace full-fidelity MC transport, we propose to break through the computational bottleneck with *Celeritas* [10], a new Graphics Processing Unit (GPU)-optimized code to run full-fidelity MC simulations of HEP detectors. General-purpose compute accelerators offer far higher performance per watt than Central Processing Units (CPUs). GPUs are the most common such devices and have become commodity hardware at the DOE LCFs and other institutional-scale computing clusters. Of the top 10 performing supercomputers in the current TOP500 list [11], all are based on accelerators and seven use NVIDIA GPUs. However, adapting scientific codes to run effectively on GPU hardware is nontrivial and indeed has been the goal of the multi-billion-dollar DOE Exascale Computing Project (ECP) [12]. The difficulty in adaptation results both from core algorithmic properties of the physics and from implementation choices over the history of an existing scientific code. The high sensitivity of GPUs to memory access patterns, thread divergence, and device occupancy makes effective adaptation of MC physics algorithms especially challenging. Existing CPU physics codes such as Geant4 are impossible to port directly to vendor-independent GPU programming models due to common C++ language

idioms such as polymorphic inheritance and dynamic memory allocation. Therefore, instead of porting existing codes, *Celeritas* is being developed from the outset.

A successful example of this approach is provided by the DOE's ECP. During the course of the ECP (2016–2023), 24 open science applications have been ported and optimized for deployment on DOE LCFs with the requirement that each meet specific performance metrics. In all cases [13], this effort required substantial code redesign involving both algorithms and data management in order to achieve performance on the GPU-based architectures that constitute the next generation of exascale computing resources at the LCFs. For most applications, this resulted in complete rewrites of the core solvers.

In this paper we present a roadmap for *Celeritas*. We describe the code architecture features, the physics and geometry implementation, and our plan to integrate *Celeritas* and DOE LCFs in current and future HEP workflows. Our primary goal is to provide a tool for efficiently using all available resources in HEP computing facilities by at least partially offloading MC production, which is one of the most intensive computing tasks in an experiment, to the network of LCFs. Although it primarily aims to reduce the computational demand of the HL-LHC, we also envision *Celeritas* being applied to the Intensity Frontier on DUNE, maximizing the use of advanced architectures that will form the backbone of high performance computing (HPC) over the next decade.

1.1 MC simulation on GPUs

The first HEP MC effort to exploit data-level parallelism, the *GeantV* project, targeted CPU vector processing units as an untapped source of computing power in deployed systems. While it demonstrated that improving data and code locality could substantially speed up the simulation software [14], it also showed that simple vectorization is insufficient to achieve the concurrency needed for transformative performance gains. Modern GPUs offer significantly more opportunities for parallelism than CPU vector processing units and have a more flexible programming paradigm that will allow performance improvements well beyond the results seen in *GeantV*.

A recently developed GPU code in the HEP MC space, Opticks, leverages the NVIDIA ray tracing library OptiX to simulate photon interactions in liquid argon detectors [15]. The extremely high (tens of millions) number of photons in flight created by a single detector interaction and relatively simple physics makes this problem an ideal candidate for GPU acceleration, and indeed the code provides a speedup in the factor of hundreds compared to transporting the emitted photons through Geant4. While Opticks was designed to take advantage of unique architectural features available on NVIDIA, the problem being solved there is highly specialized. The requirements for tracking multiple particle types through complex geometries in the presence of external magnetic fields is not well suited to the ray-tracing approach used in Opticks. Supporting Energy and Intensity Frontier experiments in HEP requires a broader set of physics and capabilities.

Shortly after the start of preliminary work on *Celeritas*, CERN launched a new effort, AdePT, to evaluate the performance potential of electromagnetic (EM) shower simulation on GPUs [16]. Initial test problem results [17, 18] have demonstrated performance parity between a 24-core Geant4 simulation and an AdePT simulation on a single consumer-level GPU. Both AdePT and *Celeritas* leverage the VecGeom geometry navigation library, a

product of the *GeantV* project designed for CPU vectorization that has since been extended to GPU multithreading using CUDA [19].

The proposed approach in *Celeritas* differs substantially from *GeantV* and AdePT. Whereas AdePT is seeking to implement GPU capability in Geant4 through minimal algorithmic and data refactoring, the *Celeritas* code architecture is designed from the outset to support algorithms and data layouts that are optimized for the unique requirements of GPU architectures. The MC particle transport method is characterized by high thread divergence, random data access patterns, high code complexity to sample physics interactions, and relatively low arithmetic intensity. Achieving high kernel occupancy to manage memory latency on GPUs requires strategies that yield local memory collocation and optimal kernel register usage [20]. These features are difficult or impossible to achieve simply by modifying code designed for cache-based architectures.

Outside the realm of HEP, recent work in the *ExaSMR: Coupled Monte Carlo Neutronics and Fluid Flow Simulation of Small Modular Reactors* project within ECP has demonstrated equivalence of 160 CPU cores per GPU on Summit[1] for the Shift MC transport code on full-featured, three-dimensional reactor models [20]. There are important differences between this work and the necessary capabilities required for particle physics detector modeling. For instance, *ExaSMR* applications are not characterized by large showers of secondary particles, and because the particles are neutral, there are no EM field interactions. Despite the more complex needs of HEP MC simulations, current preliminary results presented on this paper show a promising outcome for this project, with a fully fledged *Celeritas* potentially reaching equivalent GPU-to-CPU performance.

2 *Celeritas* code

2.1 Code Architecture

A detailed description of the *Celeritas* code architecture is given in S. R. Johnson *et al.* [21]. The code base (Fig. 1) relies on external dependencies for key capabilities that are discussed in the following sections.

Celeritas supports intra-node concurrency on multi-core architectures through OpenMP and on NVIDIA GPUs using CUDA, which we plan to supplement with a programming model for vendor-independent portability. Internode parallelism is currently implemented using the Message Passing Interface (MPI) communication library through a domain replication model in which particle events are decomposed across MPI ranks.

Like other GPU-enabled MC transport codes such as Shift [20, 22–24], the low-level component code used by transport kernels is designed so that each particle track corresponds to a single thread. There is no cooperation between individual threads, facilitating the dual host/device annotation of most of *Celeritas*. *Celeritas* also uses a modular programming approach based on composition rather than inheritance in order to accommodate device-based architectures, which have poor support for runtime polymorphism.

The *Celeritas* programming model uses the Data-Oriented Programming (DOP) paradigm [25] to facilitate platform portability, improve memory access patterns, and accelerate development. DOP separates execution code from data, and as part of this model *Celeritas*

[1] https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/

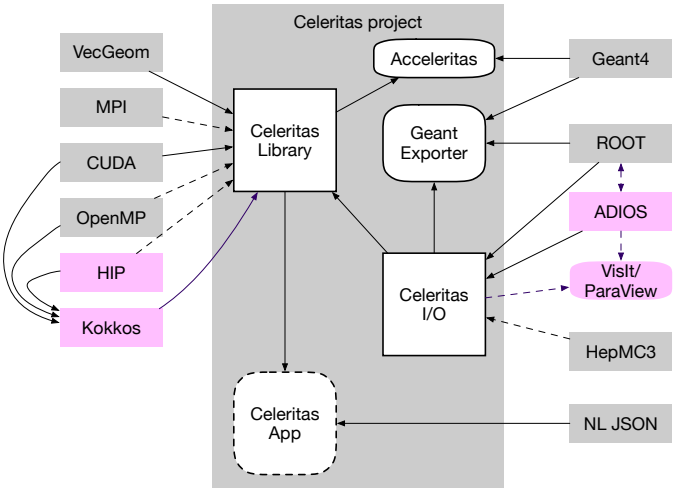


Figure 1: *Celeritas* code base (white) and its existing (gray) and proposed (magenta) third-party dependencies, both required (solid lines) and optional (dashed lines).

carefully partitions immutable, shared "parameter" data from dynamic thread-local "state" data. Object-oriented design patterns encapsulate the data storage implementation, temporarily combining parameter and state data into "view" classes. Higher-level classes use composition to combine the data from the multiple entities that comprise a particle track's complete state.

In the first 1.5 years of *Celeritas*' development, nine GPU-compatible physics models (Table 2) have been implemented. This shows DOP to be highly effective for development on heterogeneous architectures that have independent *memory spaces* between which data must be transferred. One challenge faced by MC physics application codes is the ubiquity of complicated *heterogeneous* data structures needed for tabulated physics and particle data, as opposed to the simpler *homogeneous* data layouts required by deterministic numerical solvers. A novel programming model in *Celeritas* enables the composition of new, deep data types (e.g., material properties) that are required by geometric and physics operations during the transport loop without fragmenting the underlying data layout on device.

One requirement for transporting particles in EM showers is the efficient allocation and construction of secondary particles during a physics interaction. On GPUs, managing dynamic allocations efficiently is a significant challenge. To enable runtime dynamic allocation of secondary particles, we have developed a function-like stack allocator that accesses a large on-device allocated array with a fixed capacity and uses an atomic addition to unambiguously reserve one or more items in the array. The final aspect of GPU-based secondary allocation is how to gracefully handle an out-of-memory condition without crashing the simulation *or* invalidating its reproducibility. A novel algorithm in *Celeritas* guarantees robustness when allocating secondaries, which we will extend to guarantee complete reproducibility of HEP workflow results.

The *Celeritas* code architecture just summarized is designed to enable (*i*) performance portability, (*ii*) implementation of new physics models and processes, (*iii*) optimal geometric tracking, (*iv*) optimization of the particle transport algorithm in the presence of external

EM fields, and (v) addition of scoring and User Interfaces (UI) necessary to meet HEP detector simulation requirements.

2.1.1 Platform portability

The next generation platforms at the Oak Ridge Leadership Computing Facility (OLCF), Argonne Leadership Computing Facility (ALCF), and National Energy Research Scientific Computing Center (NERSC) will each feature different GPU-based architectures as shown in Table 1, so *Celeritas* cannot rely on a single proprietary programming model. The code currently uses a limited set of macros and automatically generated kernels to support CUDA, HIP, and OpenMP. Its highly modular data management design and function-like objects used to launch kernels will allow for straightforward adaptation to other programming models as needed.

Table 1: Exascale architectures and native programming models at DOE LCFs.

Center	Machine	Integrator	GPUs	Native Programming Model
OLCF	Frontier	HPE	AMD	HIP[a]
ALCF	Aurora	Intel	Intel	OneAPI/DPC++/SYCL[b]
NERSC	Perlmutter	HPE	NVIDIA	CUDA[c]

[a] https://github.com/ROCm-Developer-Tools/HIP

[b] https://software.intel.com

[c] https://docs.nvidia.com/cuda/cuda-runtime-api/index.html

Evans *et al.* [13] surveyed the various GPU programming models employed by applications within the ECP and found Kokkos [26], HIP, and OpenMP to be the most commonly employed programming models. Each of these models has pros and cons: some models are not yet supported on all GPU architectures, and experience in ECP has shown that performance can vary dramatically depending on the maturity of both the software stack and the underlying hardware. This is also a time of significant change in the C++ language itself, particularly with respect to concurrency support via Standard Library algorithms. Combined with the increasing adoption of LLVM for C++ compiler development, we anticipate that more vendors will provide GPU-based concurrency Application Programming Interfaces (APIs) through C++ Standard Library constructs within the next 5 to 10 years, providing yet another possible means of achieving platform portability.

The *Celeritas* team will evaluate and subsequently choose a programming model to provide portable execution across all major GPU vendors as well as traditional multicore CPUs. This evaluation will be based on achievable performance across multiple GPU architectures, ease of integration into *Celeritas* workflows, sustainability, availability of multiple implementations, and ability to perform platform-specific tuning. We also plan to engage the HEP Center for Computational Excellence (CCE) Portable Parallelization Strategies (PPS) working group to ensure our strategy is in line with other efforts in the HEP community. At the present time, our nominal performance portability plan for *Celeritas* is to utilize one of the C++-based programming models (Kokkos, SYCL, or C++ standard library execution policies); however, we recognize that these models are rapidly changing as

is compiler support, and thus a proper evaluation is necessary before making a decision.

2.2 Physics

The key physics component in *Celeritas* is a *process*, which defines an observed physical phenomenon such as the photoelectric effect or bremsstrahlung. Each process is implemented as one or more models that each mathematically describe or approximate the process in a given energy regime.

The initial implementation in *Celeritas* targets EM physics between 100 eV and 100 TeV for photons, electrons, and positrons. This minimal set of capabilities, with physical processes and associated numerical models itemized in Table 2, is necessary to generate realistic simulations of EM showers and demonstrate key characteristics of a full-featured transport loop. These already implemented characteristics include (*i*) material-dependent

Table 2: Current status of *Celeritas* EM physics. The initial implementation (γ and $e^{\pm}$) is almost complete, and muon physics is in its initial stage. Particle symbols are defined in Tanabashi *et al.* [27].

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

physical properties, (*ii*) continuous slowing down in matter for charged particles, (*iii*) selecting discrete interactions among competing processes, (*iv*) scattering or absorbing particles during an interaction, (*v*) emitting secondary particles, and (*vi*) applying energy cutoffs to cull low-energy photons and electrons.

The physics implementation in *Celeritas* focuses on maximizing work done in parallel. For example, all particle types use tabulated discrete interaction cross sections calculated simultaneously in a single kernel. The primary deviation from this rule is that each model of a discrete process launches an independent kernel that applies only to tracks undergoing an interaction with that process. This set of kernel launches is performed polymorphically from CPU host code, allowing arbitrary noninvasive extensions to *Celeritas* physics.

In order to meet the detector simulation requirements for HEP experiments, *Celeritas* physics will be expanded from its initial EM prototype to a full set of particles with decay and hadronic physics. A complete list of the required physics processes and particles is shown in Table 3, where only processes are explicit as model separation will be determined based on performance and/or code maintainability.

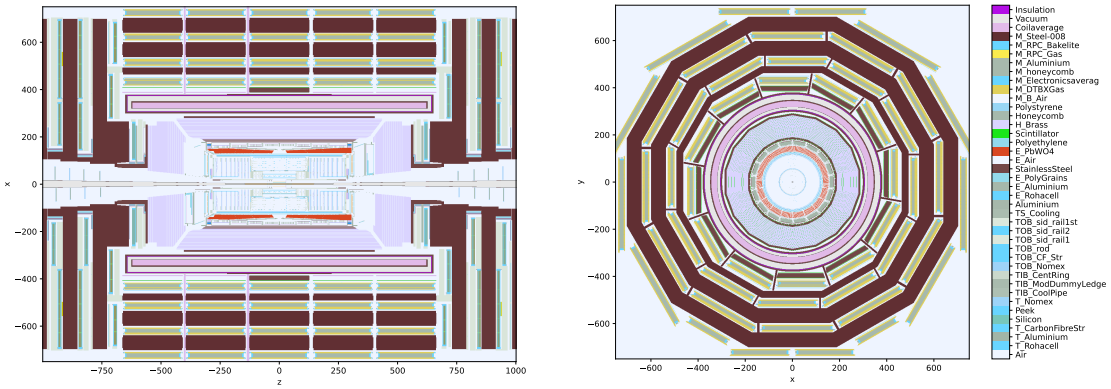
Table 3: Proposed physics development in *Celeritas*. Model definitions are ommited as these will be determined based on code performance and/or maintainability.

Physics	Process	Particle(s)
EM	photon conversion	γ
	pair annihilation	$e^{\pm}$
	photoelectric effect	γ
	ionization	charged leptons, hadrons, and ions
	bremsstrahlung	charged leptons and hadrons
	Rayleigh scattering	γ
	Compton scattering	γ
	Coulomb scattering	charged leptons, hadrons
	multiple scattering	charged leptons, hadrons
	continuous energy loss	charged leptons, hadrons, and ions
Decay	two body decay	$\mu^{\pm}$, $\tau^{\pm}$, hadrons
	three body decay	$\mu^{\pm}$, $\tau^{\pm}$, hadrons
	n-body decay	$\mu^{\pm}$, $\tau^{\pm}$, hadrons
Hadronic	photon-nucleus	γ
	lepton-nucleus	leptons
	nucleon-nucleon	p, n
	hadron-nucleon	hadrons
	hadron-nucleus	hadrons
	nucleus-nucleus	hadrons

Geant4 manages cross sections by a combination of importing experimental values from external databases and programmatic implementations of theoretical models. Most of the data used in the simulation is pre-tabulated during initialization, with some models calculating elemental cross sections at runtime. Unlike Geant4, *Celeritas* currently does not have a fully-fledged system to load and pre-tabulate cross sections from external databases. Therefore, cross section tables are imported from Geant4 via an external application—the Geant4 exporter in Fig. 1. This application loads the problem geometry via a GDML file, initializes Geant4, and stores all the pre-calculated cross sections for all the available physics processes and models into a ROOT file. This file is then used by a *Celeritas* application to load the data to host and device at initialization time.

2.3 Geometry and EM fields

Accurate simulation of HEP detector output requires a highly detailed model of the detector apparatus and nearby components. MC transport requires that the model geometry be well-defined at every point in space, requiring a model that is "watertight," or heuristics for recovering from inconsistencies (e.g., overlapping or missing regions), or both. Since particles undergo collisions and charged particles constantly change direction as they move through the magnetic fields in the detector, traditional straight-line ray tracing is necessary



(a) $y = 0$ [cm] (b) $z = 0$ [cm]

Figure 2: On-device VecGeom ray traces of the CMS generated with the *Celeritas* ray trace demonstration app.

but not sufficient to correctly navigate the geometry.

The VecGeom library supports navigation through Geant4-defined detector geometries in CPU-only code and on CUDA-enabled devices [19] and is the initial and primary geometry package in *Celeritas*. Figure 2 is a representation of the Compact Muon Solenoid (CMS) geometry traced in parallel on GPU using the *Celeritas* particle tracking interface to VecGeom.

A new, alternate geometry in *Celeritas* provides a test bed for experimenting with platform-portable navigation that uses fundamentally different algorithms from VecGeom. This implementation, Oak Ridge Adaptable Nested Geometry Engine (ORANGE), is an initial GPU port of the new modernized geometry used by the SCALE nuclear engineering code suite to model complex multi-level nuclear reactor and neutral particle shielding problems [28]. The *Celeritas* port uses the Collection paradigm to store quadric surface representations and define cell volumes as Constructive Solid Geometry (CSG) combinations of those surfaces. At present the GPU prototype implementation supports only a single geometry level, but the extension of this implementation to the full capabilities available on CPU will be relatively straightforward. We are actively collaborating with the VecGeom group at CERN to research how ORANGE and its methodology could power the next generation of Geant4 tracking on GPU.

Celeritas supports on-device propagation of particles through arbitrary magnetic fields. Using a C++-based template system for high-performance, platform-independent extensibility, it allows different integration algorithms for different user-defined fields.

2.4 User-interface and input/output (I/O)

The key challenge in providing user interfaces is developing a framework that supports integration into existing experimental MC simulation sequences that provides the minimum required code barrier for incorporation and preserves performance. Existing experimental frameworks are built on the Geant4 toolkit, which provides user actions that allow users to control the program and data flow at the level of run, event, track, and step. In Geant4

scoring is done using dedicated stepping actions in which information from the sensitive detector volumes is accessible through callback semantics into these parts of the simulation. This approach provides great user flexibility at the cost of higher computational overhead and increased system complexity. Furthermore, callback functions will not work in accelerator code because it is not possible to call host functions in the middle of device kernel execution.

Celeritas will not operate as a toolkit as Geant4 does, since this would leave many implementation decisions to the end-user, hindering performance. Therefore, to address the purely technical challenges of supporting experimental workflows, we will implement an API through which clients can specify geometric regions for scoring and MC particle data. Using this API the desired scoring data can be processed on the host at runtime, and the necessary data fields for tallies can then be configured for execution in kernel code on the device. The most efficient interface would fully occupy the device by executing many events concurrently. However, this is not the way that experimental workflows are currently configured, in which events are executed independently on each thread.

As HPC evolved to make use of heterogeneous architectures, I/O became one of the critical performance bottlenecks of many HPC applications and is a main concern for *Celeritas*. HEP detector simulations produce large volumes of data, with many millions of particle tracks and detector scoring regions having to be recorded. Thus, the need to optimize the data movement between host and device and manage parallel I/O requests is paramount. Furthermore, to be compatible with HEP experimental workflows, *Celeritas* needs to be integrated with ROOT. To address these challenges, we are collaborating with DOE's Resource and Application Productivity through Computation, Information and Data Science (RAPIDS2) team to make Adaptable Input Output System (ADIOS) the internal I/O API of *Celeritas*, as it is highly optimized for heterogeneous architectures. The ultimate goal is to find optimal strategies to mitigate I/O performance issues and integrate ADIOS with ROOT for full interoperability with HEP workflows. This collaboration with RAPIDS2 will also explore data visualization tools and event filtering, allowing users to visualize, validate, and debug the generated data before launching production campaigns.

3 *Celeritas* performance

Given the range of detector geometry complexities, primary event types, and amount of output, there is no general case that provides a simple performance number such as a simulation rate in events per second. Therefore, our ultimate performance metric will be based on a set of real-world HEP detector workflow use cases. *Celeritas* must run the targeted models at the same fidelity as the current state of the art but in much less time. In the next sections, we will present the two Figures of Merit (FOMs) that will be used to measure success, along with a few preliminary performance results comparing *Celeritas* with current state-of-the-art MC codes.

3.1 Performance metrics

On LCF hardware for the next decade, GPUs will provide the bulk of the compute capability, so one critical performance metric is the ratio of the runtime of *Celeritas* using GPUs to using only the CPUs of a given machine. This FOM represents the ability of *Celeritas* to

effectively use the hardware that is available and must be sufficiently high to justify running on LCF resources. We will target a factor of 160× for a single GPU to a single CPU, which is the relative GPU/CPU performance of the Shift MC transport code [20].

A second performance metric is critical to the programmatic viability of *Celeritas* in the broader HEP community: the work done for the same amount of cost in power consumption and hardware, using Geant4 on CPU as a baseline. This FOM is the motivation for HEP workflows to adapt to using *Celeritas* for MC transport. The Geant4 team supposes that a factor of two speedup resulting from "adiabatic improvements" to their code is not outside the realm of possibility [29], so a 2× cost improvement of *Celeritas* runtime over Geant4 runtime is our second target. Assuming that electricity consumption (and waste heat disposal) are the primary constraints for independent HEP computing centers, this factor should be evaluated by comparing the performance of Geant4 on CPUs with a comparable power requirement as *Celeritas* on GPUs. At the present time, for example, that comparison might be for an AMD 3rd Gen EPYC Processor power (280 W for 64 cores) to a PCIe NVIDIA A100 (250 W for 108 symmetric multiprocessors).

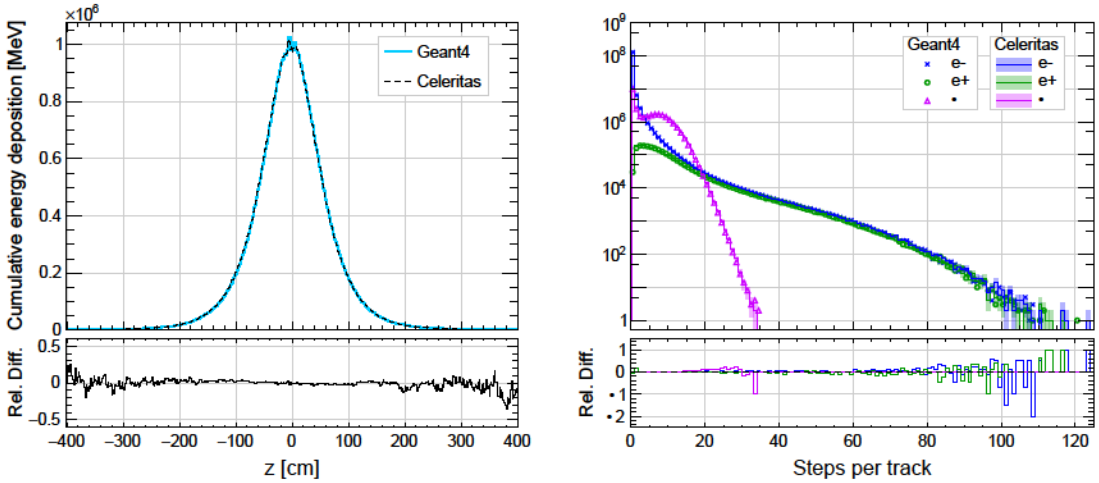
Meeting these FOMs with Geant4-equivalent physics capabilities and providing a solid user interface and I/O integration to HEP workflows will be the ultimate confirmation that *Celeritas* is a viable option for HEP experiments. At this point *Celeritas* will be ideal for execution on DOE LCFs and will be sufficiently fast on its own merits to motivate independent adoption on capacity systems.

3.2 Preliminary results

The *Celeritas* team will develop a series of internal test problems to validate every aspect of the code, including physics, geometry, EM fields, and I/O. Here, we present a small set of preliminary results comparing *Celeritas* against Geant4 using our most recently implemented high-level verification problem. This test evaluates the current code as a whole, including features such as multiple materials, secondary production, and energy cutoffs with the full array of currently implemented physics processes for γ, e^-, and e^+ (Table 2). The test geometry is an idealized coarse representation of the main components of the CMS detector, comprising six cylindrical shells centered about the z axis up to 375 cm. Each cylinder and shell is composed of a single element: vacuum, Si, Pb, C, Ti, and Fe.

The first preliminary results are on a compute node that uses the same GPUs as OLCF's Summit: it has Intel Xeon Gold 5218 (Cascade Lake) CPUs running at 2.3GHz (2 sockets of 16 cores with 2 hardware threads per core, with 188 GB total memory), and NVIDIA V100 GPUs (each with 80 symmetric multiprocessors, 64 CUDA cores per multiprocessor, and 16 GB memory). The simulation input is an isotropic 1 GeV photon point source at the origin, composed of 100 events each with 1000 photon primaries.

The initial physics validation tallies the cumulative energy deposition over the z axis (Fig. 3a), and the distribution of number of steps per track, separated by particle type (Fig. 3b). These show the two MC codes to be in agreement within statistical uncertainties. The performance numbers can only provide a baseline prediction for the eventual performance of *Celeritas*, as it has only a small subset of its eventual capabilities. Output routines are disabled in order to control for the potential latency of disk I/O, which will be improved in the future. Because the current *Celeritas* application uses the same transport loop algorithm for



(a) Cumulative energy deposition over the z axis. (b) Steps per track, separated by particle type.

Figure 3: Comparison between Geant4 and *Celeritas* simulation runs. Results are in agreement within the statistical errors.

both CPU and GPU implementations, where each step is divided into numerous kernels with many synchronization points, many tracks must be "in flight" simultaneously to amortize the overhead of thread synchronization on CPU. On GPU, one thread corresponds to a single track, but the CPU can transport an arbitrary number of tracks per physical core. This problem uses 1024 total tracks regardless of core count on CPU. For the results presented in Fig. 4, only one CPU socket (up to 16 physical cores) and a single GPU are used. Peak GPU performance is reached with about a million simultaneous tracks. At this peak, the GPU was about 30× faster than a 16-core run of the OpenMP version of *Celeritas*, the equivalent performance of 170 single-core CPU *Celeritas* runs or 240 CPUs running Geant4.

These initial results are surprisingly good for an initial unoptimized simulation of a problem with multiple realistic physics, but they do come with caveats:

- This demonstration version of *Celeritas* supports only a single element per material, and the demonstration problem is hardwired to have no magnetic field. Including those features will slow down the simulation even if they are unused.
- The pseudo-random number generator (PRNG) used in the *Celeritas* demonstration (XORWOW) is faster and less statistically random than the PRNG in Geant4 (MixMax).
- The current *Celeritas* algorithm and data structures are designed for massive GPU parallelism and are not optimal for CPU parallelism, especially with a small thread count.
- Aside from early performance analysis on a small subset of the present code base in [21], *no optimization work has been performed* for this demonstration, so *Celeritas* may become faster.
- The simulation results are reproducible with the same number of threads, but individual

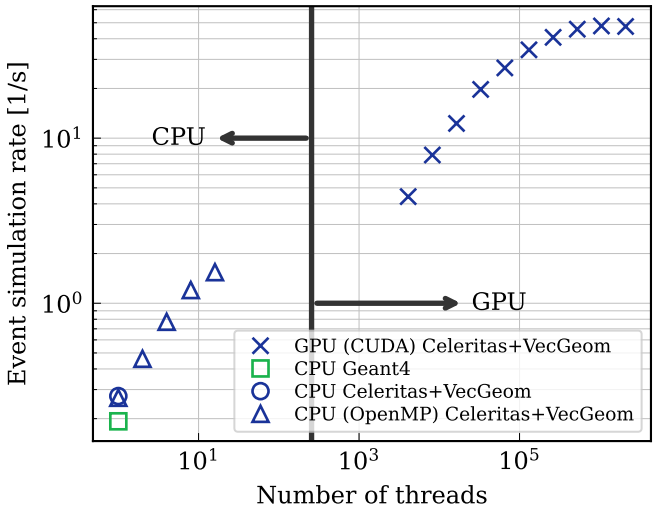


Figure 4: Performance of Celeritas on CPU and GPU with single-core Geant4 performance for reference.

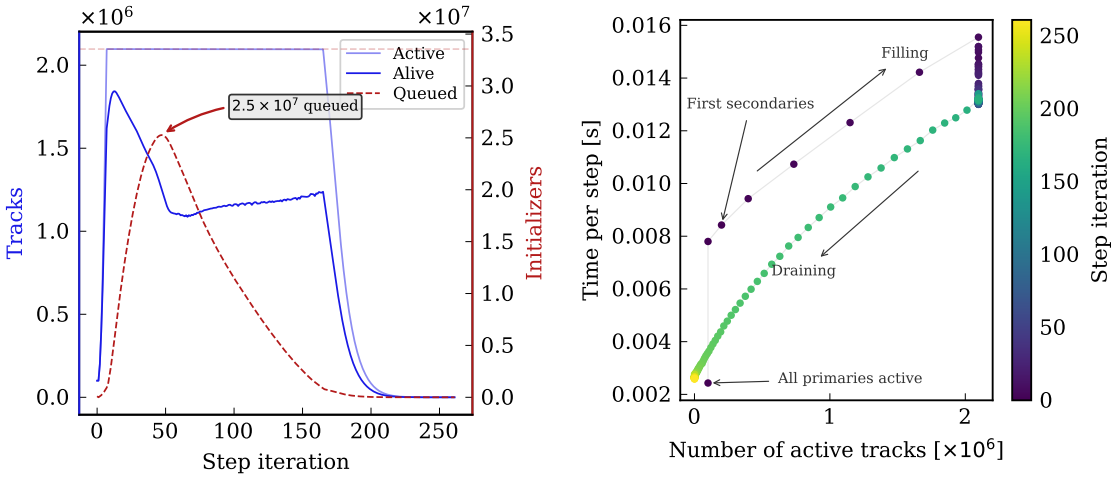
track IDs (used for "MC truth" debugging analysis) are arbitrary.

- As with all other GPU applications, good performance requires saturating the GPU with work to do, hiding latency for memory accesses and amortizing the launch cost and CPU overhead. Experimental workflows may need to batch multiple events together to achieve peak GPU performance.

Taking these considerations together, we extrapolate these initial results to be representative of expected performance on a fully featured *Celeritas* EM simulation.

Figure 5 dives into the performance characteristics of the many-threaded GPU execution by enabling extra diagnostics and detailed timing. As plotted in Fig. 5a, the behavior of the tracks in flight (blue lines, left axis) and queued secondary initializers (red line, right axis) determines the maximum thread count in the current implementation, which is limited by the memory requirements of the initializers: the current peak of 2.5×10^7 secondaries must at present be below a user-selected preallocated secondary capacity. For the first $\sim$10 steps the total particle count rises precipitously; but a lower active track count will tend to complete the transport of low-energy secondaries before starting to transport the high-energy particles that will increase the total number of tracks in flight. By preferentially selecting lower-energy particles for tracking, algorithmic improvements could reduce the total capacity requirement. Additional code improvements should allow the secondary capacity to reallocate dynamically and even buffer secondaries in CPU memory or non-volatile memory (NVM), maximizing the number of active tracks and taking full advantage of massive GPU parallelism.

The timing breakdown in Fig. 5b zooms in on time requirements of the steep rise and slow fall of active tracks. The simulation required seven step iterations to saturate the GPU with 2^{21} (about two million) active tracks from the 100 000 initial primaries. The increase in time from the initial step (all tracks encounter a boundary since they are born in a vacuum)



(a) Track and secondaries

(b) Performance per step

Figure 5: Characterization on GPU of (a) number of tracks in-flight and queued, and (b) time taken for each parallel step.

to the next step (many tracks interact in matter) is the cost of running nontrivial physics interaction kernels. Almost a third of the total iterations are to "drain" the pipeline of secondaries. Notably, at the end of the transport loop, the minimum step iteration time is still more than 10% of the maximum step time even with a handful of active tracks. This reiterates the importance of exposing parallelism for GPU performance.

With these initial results we have established a rough projection of the code's eventual performance and have begun to characterize its behavior on device as a function of workload. We will continue to use performance results to inform both low-level code optimizations (to efficiently simulate many tracks in parallel) and high-level workflow integration (to ensure enough tracks are being simulated simultaneously).

4 Integration with HEP workflows

DOE LCFs are planned to be part of HEP workflows by the scientific community, with their use ranging from simulation and reconstruction to AI methods [30]. While the Cosmic Frontier is already taking advantage of facilities such as ALCF, NERSC, and OLCF, the Energy and Intensity Frontiers have less clear integration pathways. *Celeritas* aims close the gap between HEP distributed computing networks and LCFs networks by providing three different routes (Fig. 6). These workflows intend to enable HEP experiments to use *Celeritas* in three different ways:

(a) *Acceleritas*: accelerate standard HEP detector simulation workflows built on Geant4 by offloading EM particle showers to GPUs using a new *Acceleritas* library.

(b) *End-to-end*: run complete end-to-end detector simulations with comprehensive Standard Model (SM) physics at the LCFs.

(c) *AI*: generate high-resolution detector responses as training data for AI networks to be deployed at experimental facilities as software triggers and AI-based reconstruction and event selection methods.

The next sections will discuss expected challenges and mitigation strategies, and describe these three envisioned plans to integrate *Celeritas* into experimental workflows. Since each workflow has unique traits, we expect that some experimental collaborations might see value in using all three workflows, each serving a different purpose.

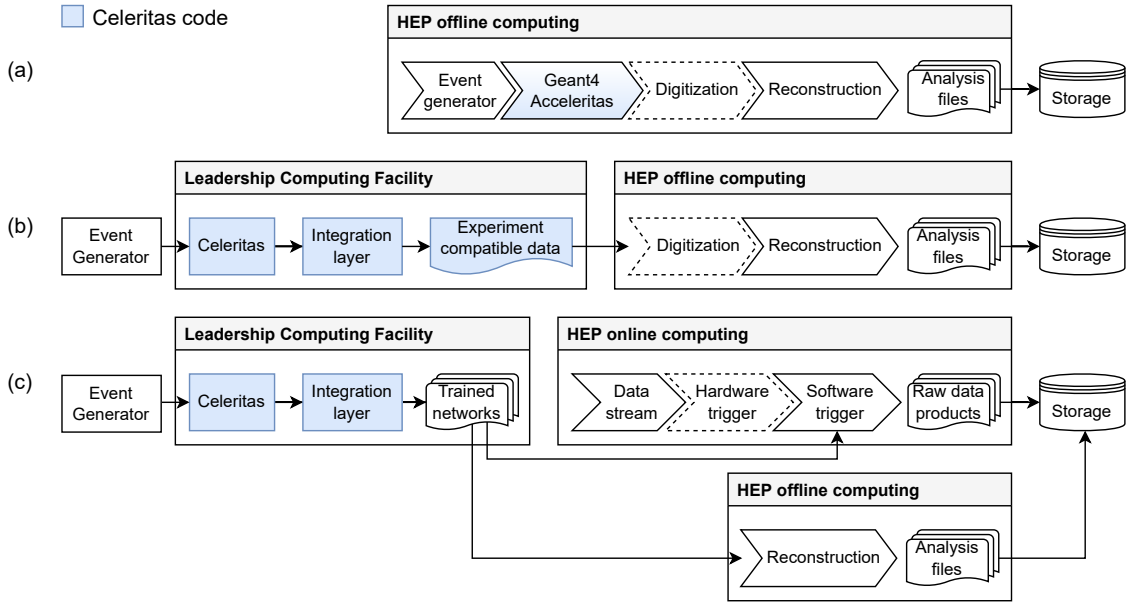


Figure 6: Proposed HEP integration workflows for (a) *Acceleritas*, (b) end-to-end *Celeritas*, and (c) *Celeritas* for AI.

4.1 Integration challenges

The heterogeneity of HEP computing workflows, associated with the volume of data produced by each experiment, pose a long list of challenges that need to be overcome in order to make *Celeritas* a viable option. We outline here the most pressing ones, along with mitigation plans.

1. Simulation inputs must be able to encompass Energy and Intensity Frontier needs. This includes user-defined geometry, physics, events, secondary particle cutoff thresholds, and sensitive detector scoring regions.
2. Output data, which entails MC particle history and detector scoring, should be flexible enough to make it compatible to experimental workflows, while maximizing I/O efficiency.
3. End-user interface must be simple enough such that the performance gain and the work needed to adapt experimental computing workflows justify the adoption.

4. Domestic and international networking between LCFs and HEP computing centers can lead to large data migration which can result in network congestions and suboptimal resource usage. The LHC's "any data, anywhere, anytime" model [30] might need special attention.

These main challenges, among other topics, will be discussed and addressed via a *Celeritas User Council*, which will be formed by members of different HEP experiments on both Energy and Intensity Frontiers. The incorporation *Celeritas* workflows into existing experimental simulation frameworks will require early engagement with the experiments. Thus, interactions with the *User Council* will determine the tradeoffs of applying *Celeritas* and LCFs to their MC production, as well as advise the *Celeritas* team when developing end-user interfaces, such that the code develops focusing on experimental compatibility.

4.2 Acceleritas

Acceleritas (Fig. 6a) is a library that will provide a hybrid workflow between *Celeritas* and Geant4. It leverages the Geant4 tasking manager system to transfer parts of the simulation to *Celeritas* for concurrent execution on device. That will allow a Geant4-based HEP detector simulation to collect a subset of particles from either primary collisions or subsequent hadronic interactions and transport them in parallel on the device using *Celeritas* while processing the remainder on the host using Geant4. The initial group of offloaded particles will be photons, electrons, and positrons, but selected hadronic physics will be conditionally integrated based on the performance of *Acceleritas*.

The Geant4 tasking manager is responsible for handling all of the major steps in the process, which includes collecting the list of particles to be offloaded, launching the *Celeritas* on-device transport loop, and merging sensitive hits and particle track data from the device back to Geant4 on the host.

Using Amdahl's law, we expect that the maximum gain of an *Acceleritas* application is $1/(1-f)$, where f is the fraction of offloaded work in a CPU-only calculation. For typical HEP events at HL-LHC and with the CMS detector geometry, the maximum speedup will be roughly a factor 3 for offloading photons and electrons to GPUs as their fractional CPU contribution is around 70%.

These gains are nowhere near our expected goals for a full end-to-end *Celeritas* simulation, where we expect *Celeritas* to achieve a similar 160× speedup factor observed in Shift. Nevertheless, *Acceleritas* will provide significant improvements while requiring minimal adaptations for current HEP workflows.

4.3 End-to-end *Celeritas*

End-to-end Celeritas integration (Fig. 6b) requires a mature *Celeritas* code, which will rely on the implementation of comprehensive SM physics capabilities along with a fully operational I/O system, including the possibility to run digitization still at the LCFs. This incorporation of *Celeritas* workflows into existing experimental simulation frameworks will require early engagement with experiments, and thus depends on a successful creation and interaction between the *Celeritas* team with members of the *User Council*.

An experimental workflow characteristic that will have to be assessed is related to how event-processing systems are handled on CPU and GPU. The most efficient interface would fully occupy the device by executing many events concurrently. However, this is not the way that experimental workflows are currently configured, in which events are executed independently on each thread. A more seamless integration of the end-to-end workflow would be to preserve independent event execution. However, this will have a dramatic effect on the achievable performance because the GPU will have to accumulate sufficient tracks to fully occupy the device.

The tradeoffs between performance and integration cost will be discussed and ultimately decided by the experimental collaborations. Finally, the volume of data transferred between LCFs and HEP computing networks is a concern that will require attention as the project evolves.

4.4 *Celeritas* for AI

AI methods are now essential in experimental HEP, with efforts to unify standard AI frameworks with HEP workflows [31]. The use of AI has been successfully deployed in detector triggering [32], training surrogate models [8], hit and image-based reconstruction algorithms [33, 34], and selection of candidate events in data analyses [35]. All of these applications require extensive generation of data for training new networks before their deployment and, as these techniques are very data and process intensive, their training can significantly impact HEP distributed computing network resources.

With increasing use of AI within HEP, we envision *Celeritas* and LCFs as tools to produce fast, full-fidelity MC samples for the purpose of training new AI networks and alleviate the workload on HEP computing centers. Standard AI frameworks already take advantage of accelerated architectures, making *Celeritas* an ideal tool for using LCFs as processing centers for AI applications in HEP.

4.5 The impact of using LCFs in HEP

The current landscape in HEP computing is evolving. Although the majority of computing resources will be CPU-based for the immediate future, GPU-accelerated hardware will be increasingly added with the advent of AI deep-learning networks in HEP data analysis and processing [36]. Thus, having MC capabilities that uses both multicore CPUs and accelerated hardware is a necessity for maximizing use of available computing resources. Combining current HEP distributed computing networks with LCFs will be sufficient to provide experiments the capability to rely completely on full-fidelity MC, instead of depending on lower-fidelity fast models due to lack of resources.

With current solutions, LHC experiments are expected to face stringent computing scenarios by the start of Run 4 [37]. A particular case is the ATLAS experiment, in which the projected available resources fall short by a factor of five [2, 9]. To show the impact that LCFs can have on experiments if fully integrated, we present a hypothetical projection comparing the compute capacity of CPU vs. GPU on ATLAS. Figure 7 plots the projected CPU total resources (blue and orange lines) and MC computing needs for Run 4 (red data point) using the data provided by Ref. [2] and converts said compute capacity to GPU (green dashed line). Here, the increase in CPU capacity is based on 10% and 20% hardware

annual improvement assuming a sustained funding, and the MC data point represents 39% of the total computing needs for ATLAS. We use the following assumptions to project the computing capacity achieved by replacing CPUs with GPUs: (*i*) the power consumption from CPU computing is converted into GPU capacity; (*ii*) the compute capability of the GPUs will increase 20% per year through a combination of increased performance and capacity—this estimated yearly capacity improvement is justified by recently observed NVIDIA performance, which has increased approximately 1.5× in the last 3–4 years; and (*iii*) each GPU is equivalent to 160 cores using the current performance achieved by Shift [20]—a performance metric that we plan to achieve with *Celeritas*.

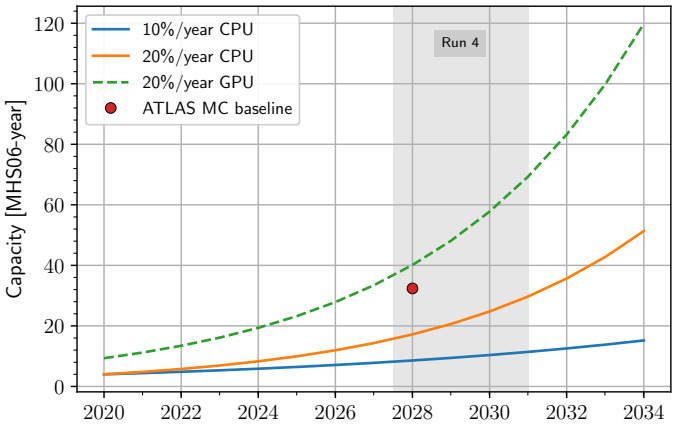


Figure 7: Compute capability converted into MHS06 year units for MC simulation. The 10% and 20% CPU capacities and 2028 MC baseline projections are the estimates for ATLAS taken directly from Ref. [2]. The green line is the compute capacity if all the current computing were converted to GPUs running at equivalent power.

The Summit supercomputer at the OLCF alone has a total of 27 648 GPUs that, using the aforementioned factor of 160, represents the equivalent of 4 423 680 CPU cores. For comparison purposes, in 2017 the Worldwide LHC Computing Grid (WLCG) encompassed approximately 500 000 CPU cores [38]. Incorporating DOE's network of LCFs will bring the total HEP computing resources to an unprecedented level and significantly alleviate current CPU bottlenecks.

5 Conclusion

We have presented *Celeritas*, a project designed to provide high-fidelity MC detector simulation transport capabilities on current and next-gen GPU architectures. Over the next years, comprehensive SM physics including EM, hadronic, and decay physics, along with EM fields will be implemented. *Celeritas*' completion is planned to happen before the beginning of HL-LHC's Run 4. This includes a fully fledged I/O system, along with its integration with ROOT, becoming a viable option to alleviate the impending MC computing requirements of the next generation of HEP experiments. With current preliminary results showing performance equivalence between a single GPU and hundreds of CPU, *Celeritas*

has the potential to bring the massive computing power provided by the DOE LCFs into HEP workflows.

The enabling technologies that will allow interfacing between end-to-end simulations performed at the LCFs and experimental computing centers will yield other long-term benefits for interactions between the DOE LCFs and experimental compute nodes. The planned I/O capabilities that need to be developed for *Celeritas* will provide full interoperability between data produced at the LCFs and ROOT, which might be benefical for other HEP code bases.

The *Celeritas* project foreshadows proposed efforts in federated computing, in which the LCFs interact directly with compute nodes at experimental facilities to provide optimal use of compute resources. In this case, we envision a scenario where expensive MC simulations are performed at the LCFs and simulation output is communicated directly to experimental HEP distributed computing networks for reconstruction and analysis. The successful execution of this project can therefore be the genesis for a host of technological advancements in the use of HPC to enable more science output in all three DOE HEP frontiers.

ACKNOWLEDGEMENTS

Work for this paper was supported by Oak Ridge National Laboratory (ORNL), which is managed and operated by UT-Battelle, LLC, for the U.S. Department of Energy (DOE) under Contract No. DEAC05-00OR22725 and by Fermi National Accelerator Laboratory, managed and operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy (Fermilab publication number for this paper is FERMILAB-FN-1159-SCD). This research was supported by the Exascale Computing Project (ECP), project number 17-SC-20-SC. The ECP is a collaborative effort of two DOE organizations, the Office of Science and the National Nuclear Security Administration, that are responsible for the planning and preparation of a capable exascale ecosystem—including software, applications, hardware, advanced system engineering, and early testbed platforms—to support the nation's exascale computing imperative.

References

[1] S. Ritz, H. Aihara, *et al.*, "Building for Discovery: Strategic Plan for U.S. Particle Physics in the global Context," Report of the Particle Physics Project Prioritization Panel (P5), Tech. Rep., May 2014. [Online]. Available: https://www.usparticlephysics.org.

[2] The ATLAS Collaboration, "ATLAS HL-LHC Computing Conceptual Design Report," CERN, CERN-LHCC-2020-015/LHCC-G-178, Nov. 2020. [Online]. Available: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults.

[3] *CMS Offline and Computing Public Results*, Dec. 2021. [Online]. Available: https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults.

[4] S. Agostinelli, J. Allison, *et al.*, "Geant4—a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, Jul. 2003. DOI: 10.1016/S0168-9002(03)01368-8. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0168900203013688 (visited on 03/23/2018).

[5] J. Allison *et al.*, "Geant4 developments and applications," *IEEE Trans. Nucl. Sci.*, vol. 53, p. 270, 2006. DOI: 10.1109/TNS.2006.869826.

[6] J. Allison, K. Amako, *et al.*, "Recent developments in geant4," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 835, pp. 186–225, 2016. DOI: https://doi.org/10.1016/j.nima.2016.06.125. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900216306957.

[7] T. Yamanaka, "The ATLAS calorimeter simulation FastCaloSim," *J. Phys. Conf. Ser.*, vol. 331, S. C. Lin, Ed., p. 032 053, 2011. DOI: 10.1088/1742-6596/331/3/032053.

[8] M. Faucci Giannelli, "FastCaloGAN: a fast simulation for the ATLAS calorimeter system using GANs," Dec. 2020. [Online]. Available: https://cds.cern.ch/record/2746428.

[9] The HEP Software Foundation, J. Albrecht, *et al.*, "A Roadmap for HEP Software and Computing R&D for the 2020s," *Computing and Software for Big Science*, vol. 3, no. 1, p. 7, Dec. 2019. DOI: 10.1007/s41781-018-0018-8. [Online]. Available: http://link.springer.com/10.1007/s41781-018-0018-8 (visited on 10/01/2019).

[10] *Celeritas*, Apache License, Version 2.0, 2021. [Online]. Available: https://github.com/celeritas-project/celeritas.git.

[11] *TOP500 LIST - NOVEMBER 2021*, Nov. 2021. [Online]. Available: https://www.top500.org/lists/top500/list/2021/11/.

[12] *The Exascale Computing Project, Addressing a national imperative: Application Development Update*, Oak Ridge, TN, Sep. 2019. [Online]. Available: www.exascaleproject.org.

[13] T. M. Evans, A. Siegel, *et al.*, "A survey of software implementations used by application codes in the Exascale Computing Project," en, *The International Journal of High Performance Computing Applications*, p. 109 434 202 110 289, Jun. 2021. DOI: 10.1177/10943420211028940. [Online]. Available: http://journals.sagepub.com/doi/10.1177/10943420211028940 (visited on 07/13/2021).

[14] G. Amadio, A. Ananya, *et al.*, *Geantv: Results from the prototype of concurrent vector particle transport simulation in hep*, 2020. arXiv: 2005.00949 [physics.comp-ph].

[15] S. Blyth, "Opticks: GPU Optical Photon Simulation for Particle Physics using NVIDIA OptiX," en, *EPJ Web Conf.*, vol. 214, A. Forti, L. Betev, *et al.*, Eds., p. 02 027, 2019. DOI: 10.1051/epjconf/201921402027. [Online]. Available: https://www.epj-conferences.org/10.1051/epjconf/201921402027 (visited on 01/22/2022).

[16] Andrei Gheata, *AdePT: Accelerated demonstrator of electromagnetic Particle Transport*, Sep. 2020. [Online]. Available: https://indico.cern.ch/event/942142/contributions/4016086/attachments/2102749/3535536/AdePT.pdf.

[17] ——, *AdePT Status*, Sep. 2021. [Online]. Available: https://indico.cern.ch/event/1052654/contributions/4525306/attachments/2310908/3932523/AdePT%2026th%20Geant4%20Collaboration%20Meeting.pdf (visited on 01/22/2022).

[18] J. Hahnfeld, *Second Look at Performance of TestEM3 in AdePT*, Apr. 2021. [Online]. Available: https://indico.cern.ch/event/1030508/contributions/4327215/attachments/2229406/3777273/TestEm3-Performance.pdf.

[19] J. Apostolakis, M. Bandieramonte, *et al.*, "Towards a high performance geometry library for particle-detector simulations," *Journal of Physics: Conference Series*, vol. 608, p. 012 023, May 2015. DOI: 10.1088/1742-6596/608/1/012023. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/608/1/012023 (visited on 12/06/2019).

[20] S. P. Hamilton and T. M. Evans, "Continuous-energy Monte Carlo neutron transport on GPUs in the Shift code," *Annals of Nuclear Energy*, vol. 128, pp. 236–247, Jun. 2019. DOI: 10.1016/j.anucene.2019.01.012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0306454919300167 (visited on 01/17/2019).

[21] S. R. Johnson, S. C. Tognini, *et al.*, "Novel features and gpu performance analysis for em particle transport in the celeritas code," *EPJ Web Conf.*, vol. 251, p. 03 030, 2021. DOI: 10.1051/epjconf/202125103030. [Online]. Available: https://doi.org/10.1051/epjconf/202125103030.

[22] T. M. Pandya, S. R. Johnson, *et al.*, "Implementation, capabilities, and benchmarking of Shift, a massively parallel Monte Carlo radiation transport code," *Journal of Computational Physics*, vol. 308, pp. 239–272, Mar. 2016. DOI: 10.1016/j.jcp.2015.12.037. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0021999115008566 (visited on 06/19/2017).

[23] S. P. Hamilton, S. R. Slattery, *et al.*, "Multigroup Monte Carlo on GPUs: Comparison of history- and event-based algorithms," en, *Annals of Nuclear Energy*, vol. 113, pp. 506–518, Mar. 2018. DOI: 10.1016/j.anucene.2017.11.032. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0306454917304231 (visited on 12/28/2017).

[24] S. P. Hamilton, T. M. Evans, *et al.*, "Domain decomposition in the GPU-accelerated Shift Monte Carlo code," en, *Annals of Nuclear Energy*, vol. 166, p. 108 687, 2022. DOI: 10.1016/j.anucene.2021.108687.

[25] Yehonathan Sharvit, *Data-Oriented Programming: Unlearning Objects*. Manning Publications Co., 2022.

[26] H. C. Edwards, C. R. Trott, *et al.*, "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns," *Journal of Parallel and Distributed Computing*, vol. 74, no. 12, pp. 3202–3216, 2014, Domain-Specific Languages and High-Level Frameworks for High-Performance Computing. DOI: https://doi.org/10.1016/j.jpdc.2014.07.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731514001257.

[27] M. Tanabashi, K. Hagiwara, *et al.*, "Review of Particle Physics," en, *Physical Review D*, vol. 98, no. 3, p. 030 001, Aug. 2018. DOI: 10.1103/PhysRevD.98.030001. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.98.030001 (visited on 08/05/2020).

[28] W. A. Wieselquist and L. R. A., "SCALE 6.3.0 User Manual," ORNL, Oak Ridge, TN, USA, Tech. Rep. ORNL/TM-SCALE-6.3.0, Dec. 2021.

[29] Marc Verderi and Alberto Ribon, *Geant4 Introduction*, Apr. 2021. [Online]. Available: https://indico.cern.ch/event/1028379/ (visited on 01/21/2022).

[30] J. Zurawski, B. Brown, *et al.*, "High energy physics network requirements review (final report, july-october 2020)," Jun. 2021. DOI: 10.2172/1804717. [Online]. Available: https://www.osti.gov/biblio/1804717.

[31] L. Giommi, V. Kuznetsov, *et al.*, "Machine Learning as a Service for High Energy Physics on heterogeneous computing resources," *PoS*, vol. ISGC2021, p. 019, 2021. DOI: 10.22323/1.378.0019.

[32] T. M. Hong, B. Carlson, *et al.*, "Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics," *JINST*, vol. 16, no. 08, P08016, 2021. DOI: 10.1088/1748-0221/16/08/P08016. arXiv: 2104.03408 [hep-ex].

[33] S. Bhattacharya *et al.*, "GNN-based end-to-end reconstruction in the CMS Phase 2 High-Granularity Calorimeter," Mar. 2022. arXiv: 2203.01189 [physics.ins-det].

[34] L. de Oliveira, M. Kagan, *et al.*, "Jet-images — deep learning edition," *Journal of High Energy Physics*, vol. 2016, no. 7, Jul. 2016. DOI: 10.1007/jhep07(2016)069. [Online]. Available: http://dx.doi.org/10.1007/JHEP07(2016)069.

[35] P. Adamson *et al.*, "Constraints on Oscillation Parameters from ν_e Appearance and ν_μ Disappearance in NOvA," *Phys. Rev. Lett.*, vol. 118, no. 23, p. 231 801, 2017. DOI: 10.1103/PhysRevLett.118.231801. arXiv: 1703.03328 [hep-ex].

[36] A. Radovic, M. Williams, *et al.*, "Machine learning at the energy and intensity frontiers of particle physics," en, *Nature*, vol. 560, no. 7716, pp. 41–48, Aug. 2018. DOI: 10.1038/s41586-018-0361-2. [Online]. Available: http://www.nature.com/articles/s41586-018-0361-2 (visited on 09/14/2021).

[37] *Longer term LHC schedule*, Jun. 2021. [Online]. Available: https://lhc-commissioning.web.cern.ch/schedule/LHC-long-term.htm.

[38] A. Valassi, M. Alef, *et al.*, "Using HEP experiment workflows for the benchmarking and accounting of WLCG computing resources," *EPJ Web of Conferences*, vol. 245, C. Doglioni, D. Kim, *et al.*, Eds., p. 07 035, 2020. DOI: 10.1051/epjconf/202024507035. [Online]. Available: https://www.epj-conferences.org/10.1051/epjconf/202024507035 (visited on 01/21/2022).