



AdePT

Accelerated demonstrator of electromagnetic Particle Transport

Andrei Gheata for the AdePT Developers

Compute Accelerator Forum - June 29, 2022

Targets



- ▶ **Functionality:** make all simulation components work on GPU
 - Physics, geometry, field, but also scoring code to limited extent
 - Prototype e^+ , e^- and γ EM shower simulation on GPU
- ▶ **Correctness:** validate results and ensure reproducibility
 - Against Geant4 equivalent
- ▶ **Usability:** integrate in a hybrid CPU-GPU Geant4 workflow
 - For realistic experimental setups
- ▶ **Performance:** understand/address bottlenecks limiting performance
 - Estimate feasibility and effort for efficient GPU simulation

The project

- ▶ GitHub [repository](#)
 - Initial commit in Sep 2020, $\mathcal{O}(10)$ contributors
- ▶ Strategy: integrate gradually features as new examples
 - No library build, maximize flexibility to explore different directions
- ▶ Few external dependencies
 - Geometry: [VecGeom](#) library, enhancing GPU-related features
 - Physics: [G4HepEm](#) library, a GPU-friendly port of Geant4 EM interactions
- ▶ Portability aspects not a major priority in this project phase
 - Preliminary investigations started with Alpaka and OneAPI

apt-sim / AdePT Public

Code Issues Pull requests Discussions Actions Projects Wiki Security Insights

Filters: is:issue is:open Labels: 23

7 Open 28 Closed Author Label Projects Milestones Assignee Sort

- Validation and benchmarking of latest examples with single-precision geometry **performance**
#155 opened 21 days ago by aghasta
- Integration of AdePT/Geant4 fast-sim based interface with the latest state of the art AdePT example **priority**
#151 opened on Oct 19 by WitakPolanski
- Support for non-constant magnetic field **priority**
#150 opened on Oct 19 by WitakPolanski
- Optimize field propagation with safety **performance**
#149 opened on Oct 19 by hatrop
- Field propagator uses wrong path length **bug**
#148 opened on Oct 15 by hatrop
- Think about memory coalescing **performance**
#157 opened on Jan 20 by hatrop
- Deal with memory limitations on the device **performance**
#156 opened on Jan 19 by hatrop

ProTip! Notify someone on an issue with a mention, like: @agheats.

The prototype

Initial commit

Fisher-Price like example & Alpaka version of it

Simple workflow with geometry navigation (exa2)

First example working in constant field (exa4)

First integration with G4HepEm physics (exa5)

First example with geometry and Bz field (exa6)

Added gamma interactions (exa9), added TestEm3 similar to Geant4

Added import/export of geometry and physics between Geant4 and AdePT (exa7)

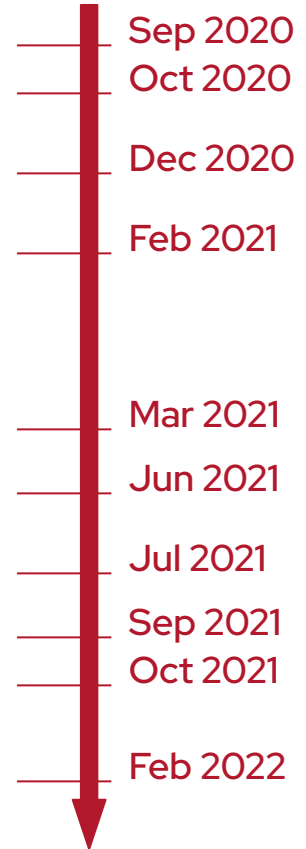
MT version of TestEm3

Single-precision support in geometry

Standalone example with generalized GDML geometry (exa13)

Added support for multiple scattering

Integration with Geant4 workflow demonstrator (exa14)



GPU-friendly rewrite of EM physics



- ▶ **G4HepEm: compact** library of EM processes for HEP
 - Covers the complete physics for e^- , e^+ and γ particle transport
 - Initialization of physics tables dependent on Geant4, but usage on GPU standalone and lightweight
- ▶ Design of library very supportive for **heterogeneous** simulations
 - Interfaces: standalone functions without global state
 - Data: physics tables and other data structures copied to GPUs
 - Reusing > 95% of the code from G4HepEm for GPU shower simulation

Interactions modelled for e^+ , e^- and γ

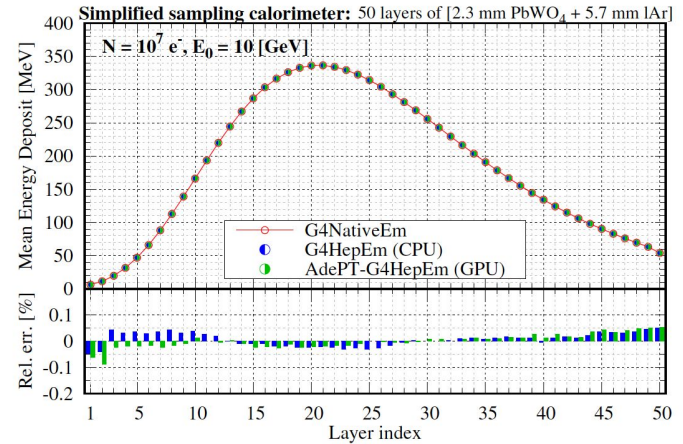
Particle	Interactions	Models	Geant4 (EM-Opt0)	G4HepEm (with G4HepEm prefix)	Energy Range
e^-	Ionisation	Moller	G4MollerBhabhaModel	ElectronInteractionIoni	1 keV - 100 TeV
	Bremsstrahlung	Seltzer-Berger	G4SeltzerBergerModel	ElectronInteractionBrem (including both models)	1 keV - 1 GeV
		Rel. model ¹	G4eBremsstrahlungRelModel		1 GeV - 100 TeV
	Coulomb scat. ²	Urban	G4UrbanMscModel	ElectronInteractionUMSC	1 keV - 100 MeV
Wentzel-VI		G4WentzelVIModel	100 MeV - 100 TeV		
e^+	Ionisation	Bhabha	G4MollerBhabhaModel	ElectronInteractionIoni	1 keV - 100 TeV
	Bremsstrahlung	Seltzer-Berger	G4SeltzerBergerModel	ElectronInteractionBrem (including both models)	1 keV - 1 GeV
		Rel. model	G4eBremsstrahlungRelModel		1 GeV - 100 TeV
	Coulomb scat.	Urban	G4UrbanMscModel	ElectronInteractionUMSC	1 keV - 100 MeV
		Wentzel-VI	G4WentzelVIModel		100 MeV - 100 TeV
Annihilation	$e^+ - e^- \rightarrow 2\gamma$	G4eplusAnnihilation	PositronInteractionAnnihilation	0^3 - 100 TeV	
γ	Photoelectric	Livermore	G4LivermorePhotoElectricModel	GammaInteractionPhotoelectric ⁴	0^5 - 100 TeV
	Compton scat.	Klein - Nishina ⁶	G4KleinNishinaCompton	GammaInteractionCompton	100 eV - 100 TeV
	Pair production	Bethe - Heitler ⁷	G4PairProductionRelModel	GammaInteractionConversion	$2m_0c^2$ - 100 TeV
	Rayleigh scat.	Livermore	G4LivermoreRayleighModel	not considered to be covered at the moment	100 keV - 100 TeV

*Energy loss fluctuation⁻ corresponding to G4UniversalFluctuation model in Geant4-11.p01 also implemented for e^+ , e^-

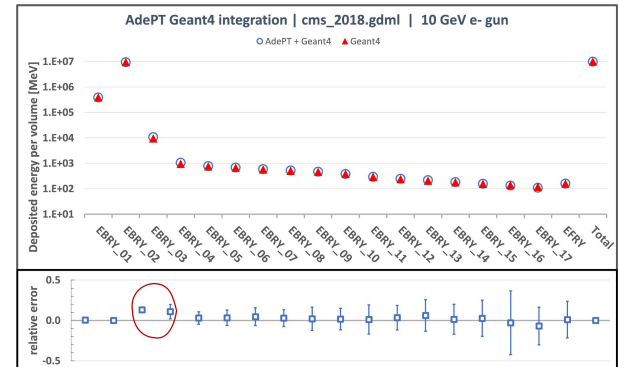
Correctness checks

- ▶ Validation against Geant4 standalone is essential
 - Comparisons to CPU references (in general Geant4-based) done for each added functionality
 - Both for standalone and Geant4 integration examples
- ▶ EM physics now fully validated
 - At ‰ level in the sampling calorimeter test case
- ▶ Still working on the last bugs/features in a **hybrid** workflow steered by Geant4

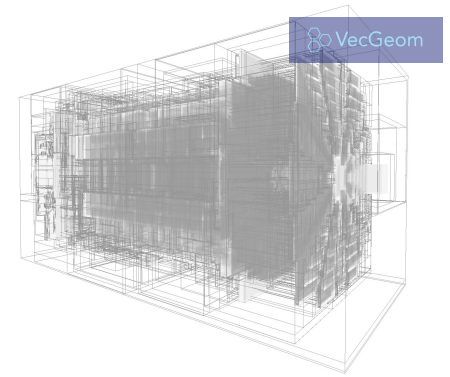
Sampling calorimeter example



AdePT integration with Geant4



GPU geometry: VecGeom




- ▶ First implementation of GPU support few years old
 - C++ types re-compiled using *nvcc* in a separate namespace/library
 - In AdePT we wrote a custom global navigation layer calling lower level VecGeom APIs
- ▶ Improving gradually GPU support
 - Developed custom optimised navigation state, single-precision support
 - Moving from a simple “loop” navigator to an optimized **BVH navigator**
 - Adopting modern CMake GPU support
- ▶ Moving forward: **specializing** the VecGeom GPU navigation support
 - Portable less complex code, creating a surface-based view on device
 - An initial prototype is now being discussed

Parallelization in AdePT

- ▶ Simulation is done in steps, moving particles to either boundaries or physics processes
- ▶ All active tracks available are stepped at once (Geant4 transports one particle at a time)
 - Much higher degree of parallelism and more uniform work for the GPU
- ▶ No “thread-local” state, everything embedded in the track
 - Energy, position/direction, state needed across steps
 - Random number generator state (RANLUX++) per track to ensure reproducibility
 - ▷ Strategy to spawn a new sequence for daughter particles from the current state
- ▶ Tracks pre-allocated per particle type in thread-safe containers
 - Atomic counter to hand on track slots to be filled by kernels (explained later)

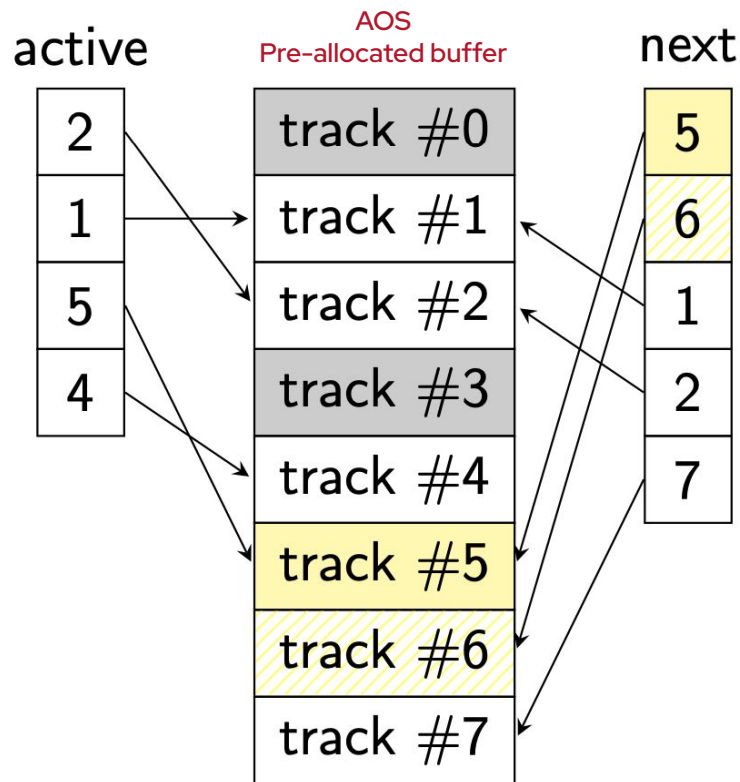
Track representation / access pattern



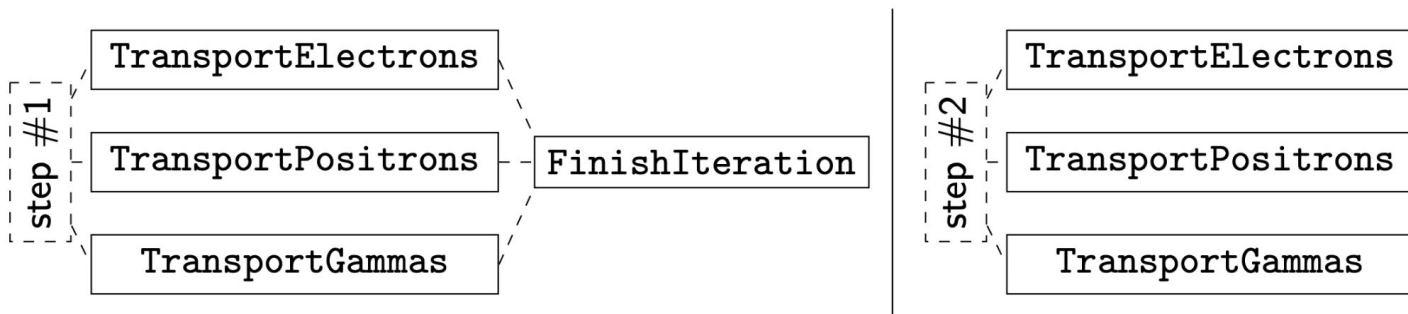
- ▶ Investigated entity component systems approach used in gaming (SoA)
 - Now also investigating track data structure transformations using 
 - No definitive conclusion yet
 - ▷ In realistic setups track data access is just a fraction of the loads/stores
 - ▷ Making small kernels accessing just part of the data introduces other overheads
- ▶ Difficult to implement coalesced memory access in simulation
 - Complex and sparse (accessed) data models. Geometry is a pathologic case.
 - The **stochastic** nature of the problem destroys locality.
 - Killed tracks leave random holes in the track data structure.
 - ▷ Placing statically data makes accessing it look like a “whack-a-mole” game
 - ▷ We need data regrouping solutions

Handling arrays of tracks in AdePT

- ▶ Store indices of active tracks (per particle type)
 - Parallelize transportation kernels over these indices
- ▶ Queue indices for “next” active tracks
 - Both secondaries and “surviving” tracks
 - Implemented with atomic counter
- ▶ Run transportation kernel stepping the active tracks
 - Here track #1 , #2 and #5 survive, track #4 dies, and track #6 and #7 are produced
- ▶ Swap active ↔ next before next iteration
 - Compacting unused slots now possible



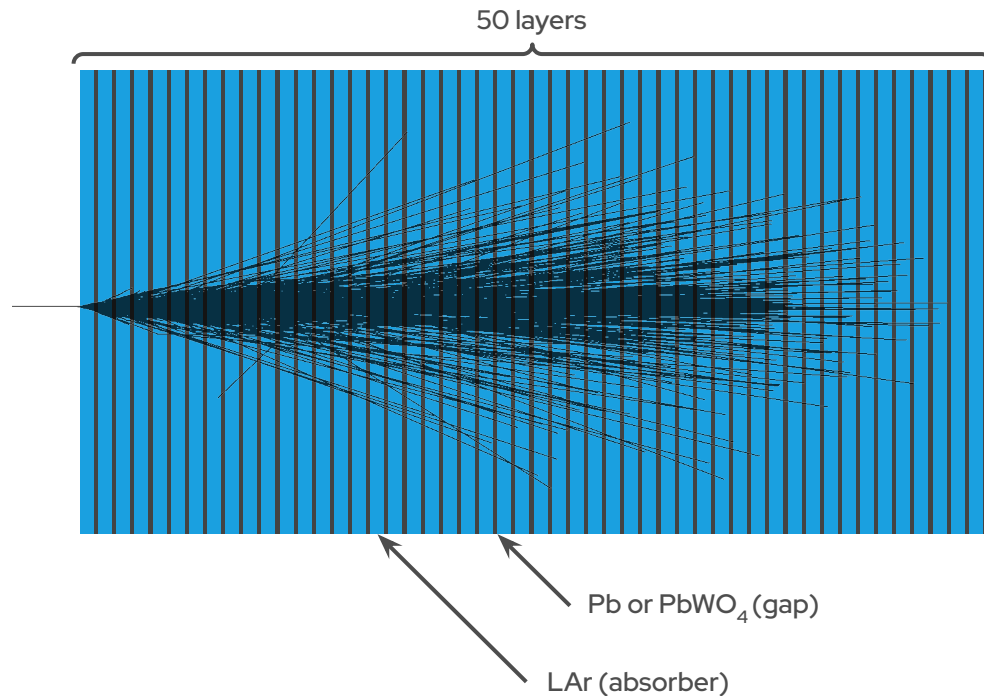
Stepping workflow, a first approach



- ▶ Can start kernels for particle types in parallel streams (transport is independent)
- ▶ Synchronization means overhead
 - Synchronize on the GPU via CUDA events
 - Synchronize with host once at the end of the step (stepping loop control, transfer hits)
- ▶ Main optimization playground
 - Better work balancing between warps, reducing impact of tails, better device occupancy

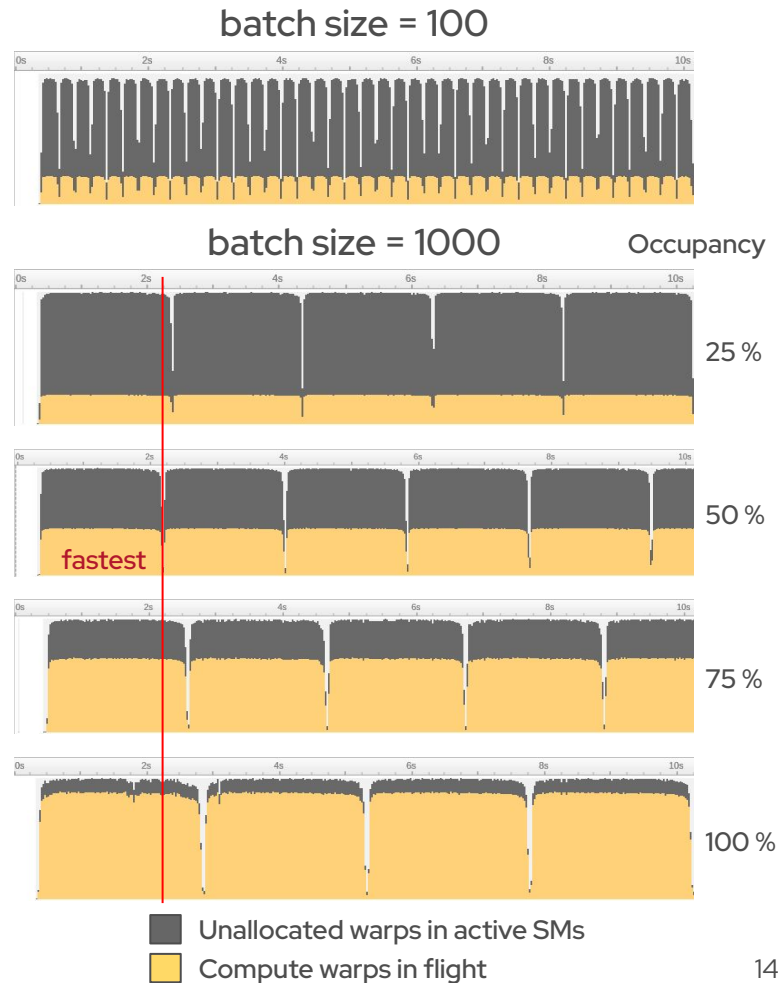
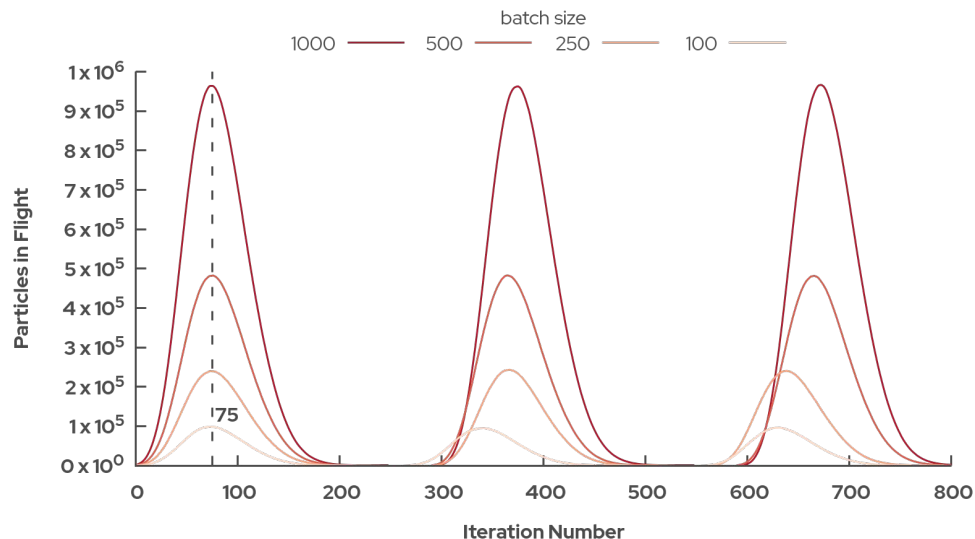
Simplified Calorimeter Benchmark

- ▶ Calorimeter with 50 layers
- ▶ Simulate 10,000 particles
 - 10 GeV electrons as primaries
- ▶ Configuration parameter space
 - Number of particles per batch
 - Number of registers per thread
 - Number of threads per block
- ▶ Compare on different hardware
 - Nvidia RTX 2070
 - Nvidia RTX 8000
 - Nvidia Tesla V100S



Run Time Characteristics

- putting more work per batch does more work in the same #iterations (steps)
 - limited by available memory AND available tracks
- hints already to using strategies to fill the gaps
 - e.g. more CPU threads doing concurrent events



Kernel Launch Configurations

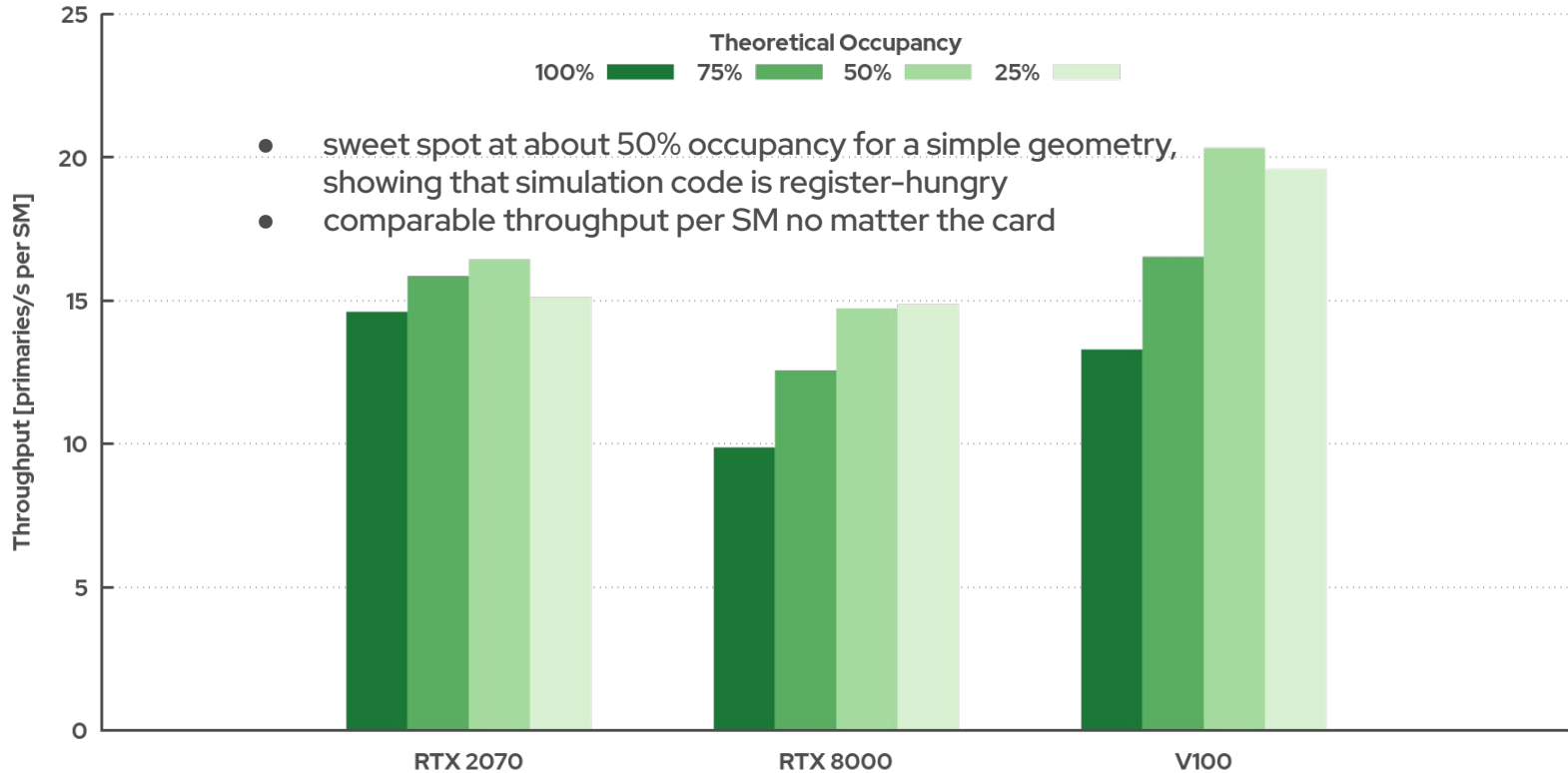
- ▶ 1024 Threads / SM
 - 4 schedulers x 8 warps/scheduler x 32 threads/warp
- ▶ 65536 Registers / SM
 - 4 register files x 16384 registers
 - 1 float = 1 register, 1 double = 2 registers
- ▶ 96 KB L1 Data Cache / Shared Memory
- ▶ Theoretical Occupancy (`-maxrregcount` or `__launch_bounds__`)
 - 256 regs/thread (256 threads, 8 warps) \Rightarrow 25%
 - 160 regs/thread (320 threads, 10 warps) \Rightarrow 38%
 - 128 regs/thread (512 threads, 16 warps) \Rightarrow 50%
 - 96 regs/thread (640 threads, 20 warps) \Rightarrow 63%
 - 80 regs/thread (768 threads, 24 warps) \Rightarrow 75%
 - 64 regs/thread (1024 threads, 32 warps) \Rightarrow 100%

Higher parallelism
Faster Threads

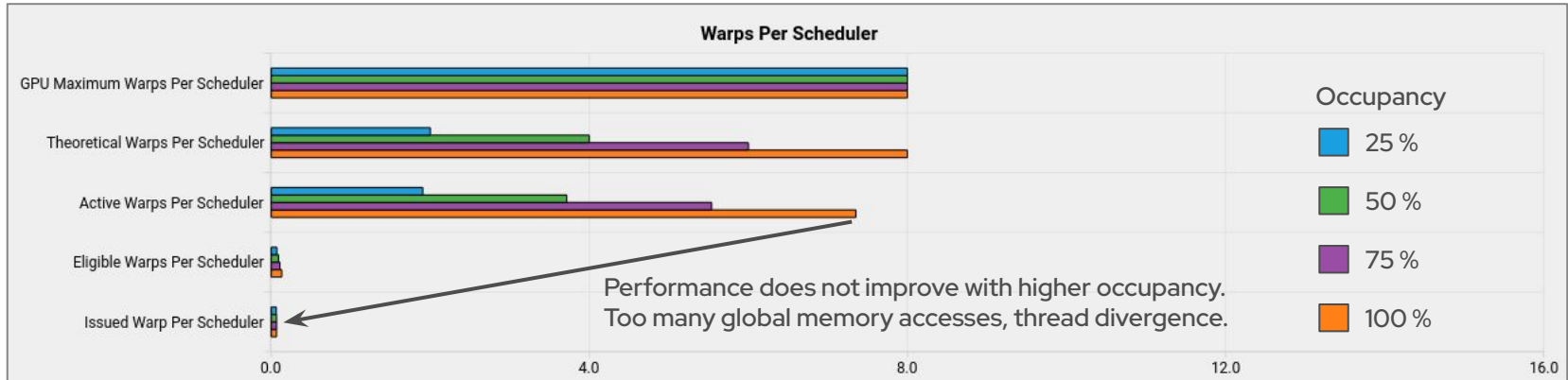
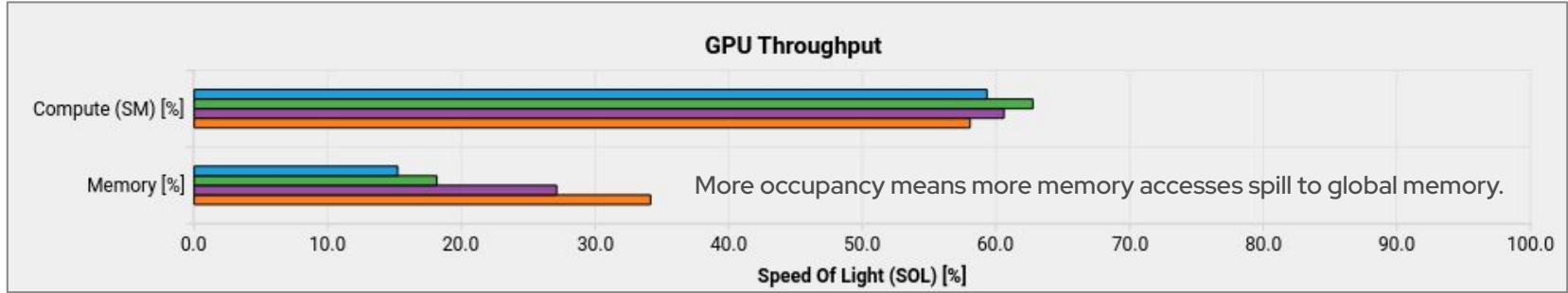
Turing SM



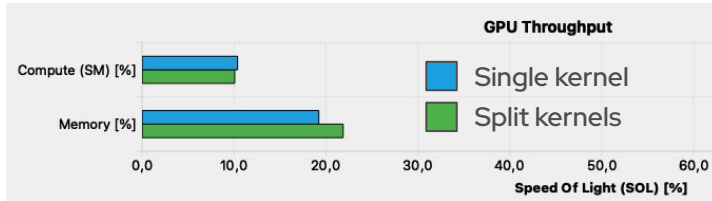
Relative Performance per SM



GPU Throughput (RTX 2070)



Case Study: Thread Divergence

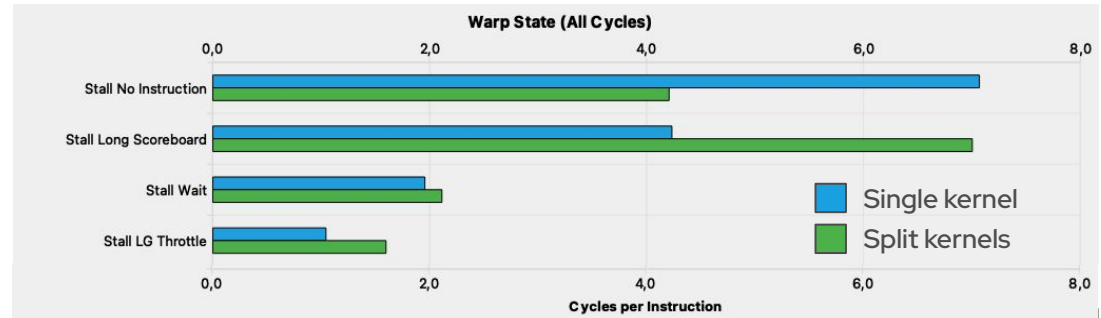
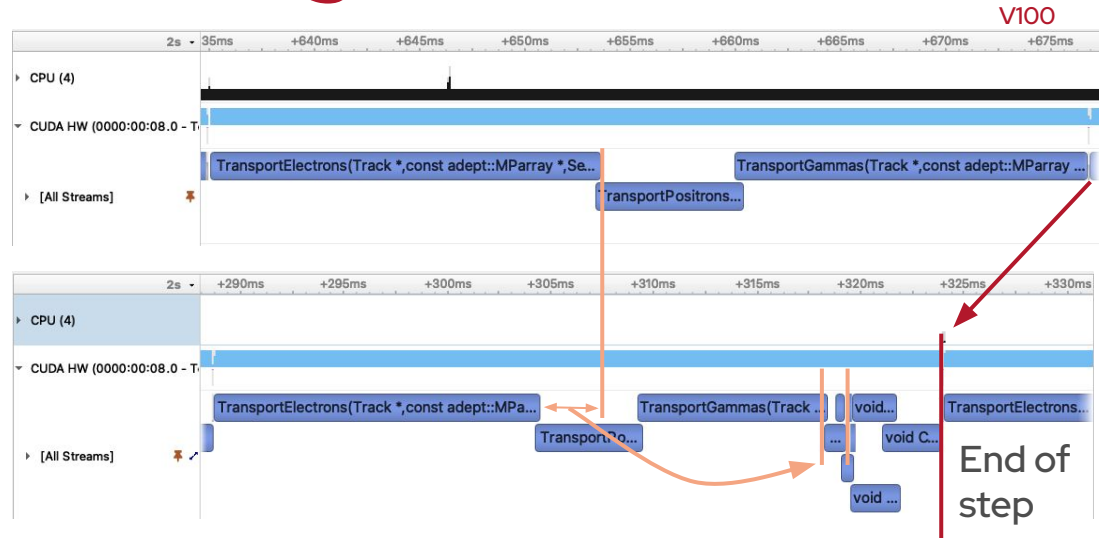


Problem: Threads in transport kernels diverge because of diverging interactions
 → 13 / 32 threads active on average

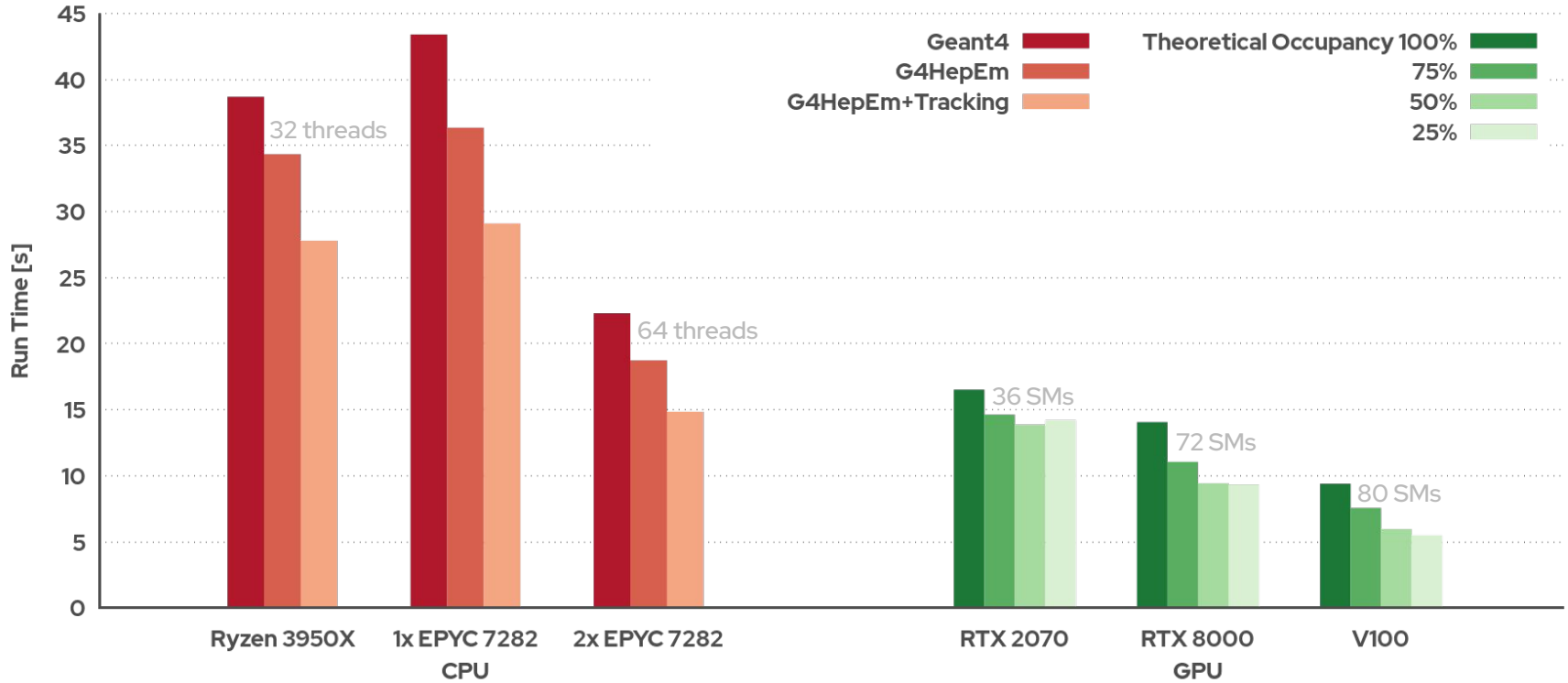
Here: Split off interaction computations from cross-section and geometry kernels (one kernel for pair creation, one for ionisation, ...)

Result: 17 / 32 threads active for physics + geo
 29 / 32 threads active for Bremsstr.
 Run time: 6.4 s → 5.5 s

Conclusion: Keeping threads coherent is key for detector simulation
 Generally difficult; stochastic processes



CPU vs GPU Performance



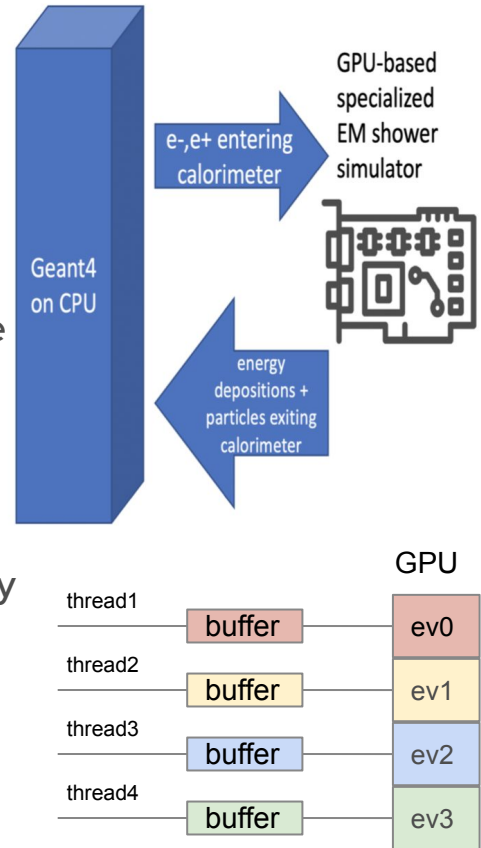
AMD Ryzen 3950X (16 cores, 32 threads, 3.5-4.7GHz), AMD EPYC 7282 (16 cores, 32 threads, 2.8-3.2GHz)

Performance portability

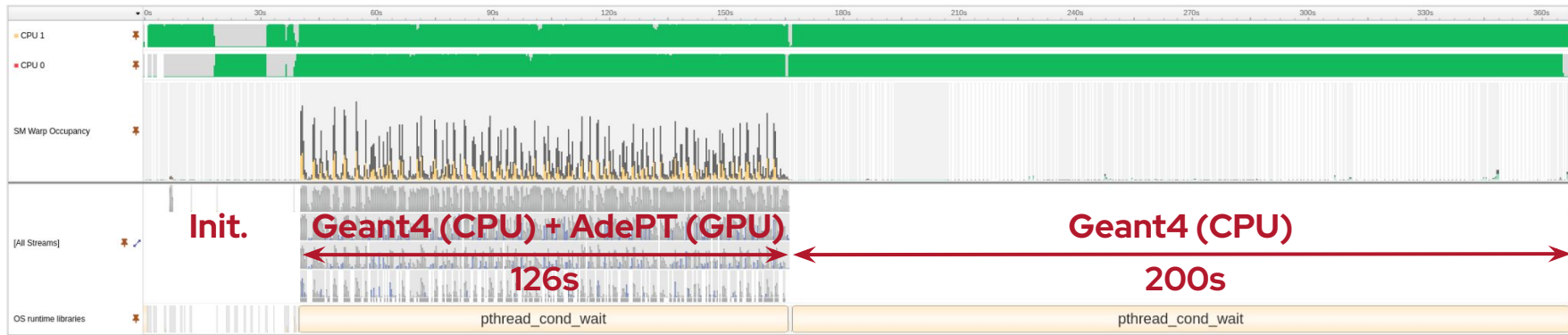
- ▶ oneAdePT – port to oneAPI of an AdePT snapshot
 - core utilities, magnetic field, RNG, G4HepEM
 - No way around calling legacy CUDA code compiled in VecGeom
- ▶ Many obstacles for migrating CUDA to DPC++ code
 - SYCL limitations in calling virtual functions or function pointers, non-const globals, support for `std::` math functions, support for CUDA compiled libraries, documentation
- ▶ Triggered investigations and work in VecGeom
 - Non-virtual dispatch and CUDA compilation using clang, deeper restructuring needed
 - Specializing geometry for GPU needed for both portability and better performance
 - Further efforts for portability postponed until solving this blocker

AdePT-Geant4 integration

- ▶ AdePT only provides EM physics for e^+ , e^- and γ
 - Cannot be used standalone for simulating a full experiment
 - In a first phase it could be used as accelerator for the EM part, in the same way as fast simulation models can be used in Geant4
- ▶ Developed an integration interface allowing a Geant4 region to become the “GPU region”
 - Intercepting and buffering for GPU particles sent asynchronously by Geant4 threads
 - May be in future applicable to the full detector, handing produced hadrons back to Geant4



CMS Simulations: Integrated and Standalone

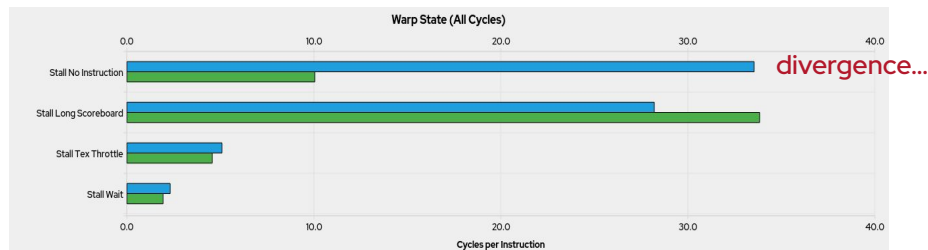
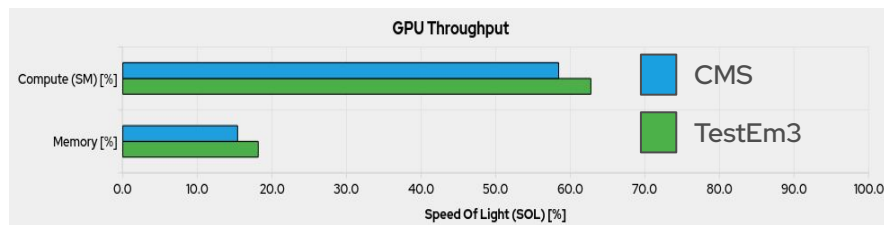


Integrate into Geant 4?

Above is a timeline of a simulation of CMS comparing the AdePT integration and Geant4 (Ryzen 3950X, RTX2070), with a **speedup of 37%** when using 2 CPU threads + 1 GPU vs only 2 CPU threads.

Impact of detector geometry?

On the right, 10^6 electrons at E=10 GeV on an Nvidia Tesla V100 with TestEm3 geometry vs the CMS geometry. The total simulation run time for the simplified calorimeter (TestEm3) setup is **549s** vs **1455 s** for the CMS geometry (a slowdown of 2.65x).



User actions & scoring

- ▶ Geant4 calls user code for performing custom run, event and stepping actions
 - Should we provide the same for simulation running on GPU?
 - Do we have to run (complex) user code there?
 - ▷ Code efficiency, device data management, transfer to host
- ▶ Solving this was not an immediate priority
 - First target: EM calorimeters, allowing for pre-defined scoring type
 - However simple energy deposits are not enough in several use cases
- ▶ A simplified approach based on static binding possible at this stage (we compile the transport kernels)
 - Init on device, score energy deposits, copy hits to host, clear
 - Called for sensitive detectors within the device stepping loop

```
SimpleScoring.h  
  
struct SimpleS  
{  
    BasicScoring *InitializeOnGPU();  
  
    __device__ void Score(params);  
  
    template <typename Stream>  
    void CopyHitsToHost(Stream  
        &stream)  
    void ClearGPU(Stream &stream)  
};  
  
using AdeptScoring = SimpleS;
```

```
electrons.cuh  
  
template <typename Scoring>  
__global__ void  
TransportElectrons(Scoring *s)  
{  
    ...  
    s->Score(track_state_pars);  
}
```

Outlook

- ▶ A challenging project, the problem is far from a perfect match for GPU
 - Fast progress due to some code refactoring done before AdePT (VecGeom, field)
 - ▷ Re-writing these is now necessary due to performance reasons
 - Several performance limitations in the path still to be addressed, some require deep code restructuring
- ▶ Prototypes for standalone and Geant4-integrated workflows available
 - Realistic examples for LHC setups, GPUs can be used in a Geant4 native application
 - Optimization work ongoing, performance not yet on a GPU-efficient baseline
- ▶ Most initial AdePT objectives complete
 - Still to decide on the strategy for larger developments and more efficient integration with the experiment's simulation code
- ▶ Collaborating on common development topics with the Orange team is essential: geometry, integration with Geant4 and experiments code, ...