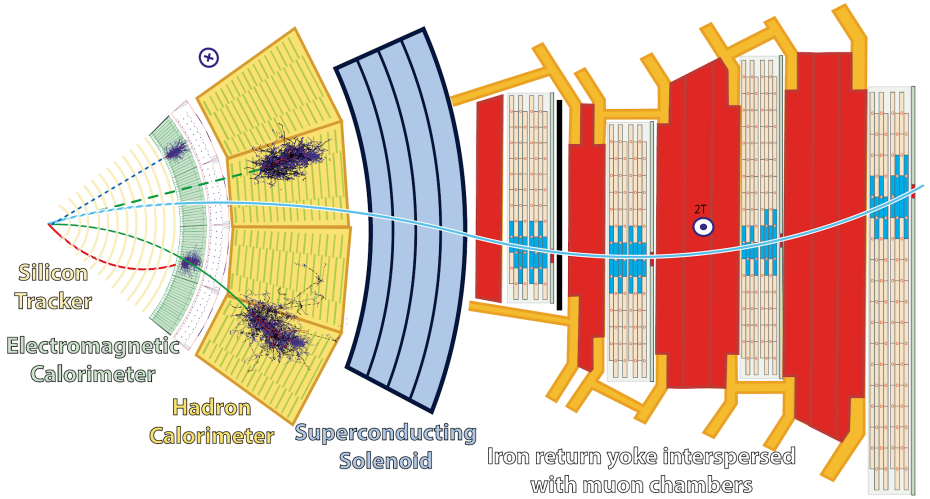
AdePT

accelerating GEANT4 applications on GPUs

Severin Diederichs on behalf of the AdePT team

GEANT4 – a mini introduction

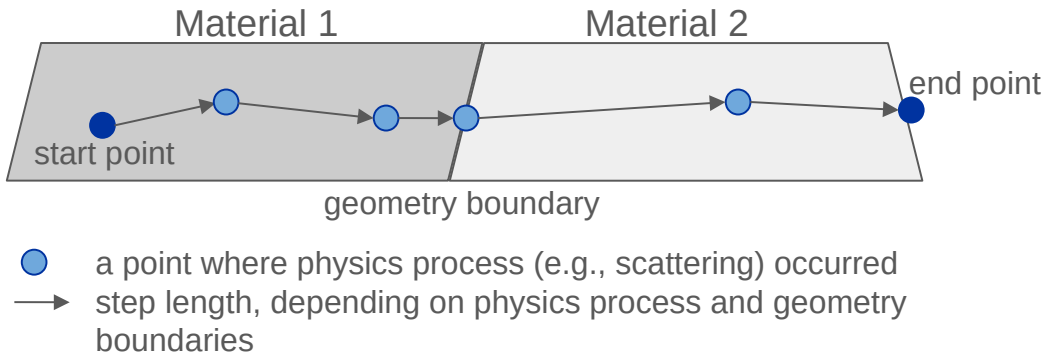
Monte-Carlo particle transport code. Backbone of detector simulation in HEP.



Also used in medical physics, space physics, accelerator physics, radiation protection…

GEANT4 – a mini introduction

Monte-Carlo particle transport code. Backbone of detector simulation in HEP.



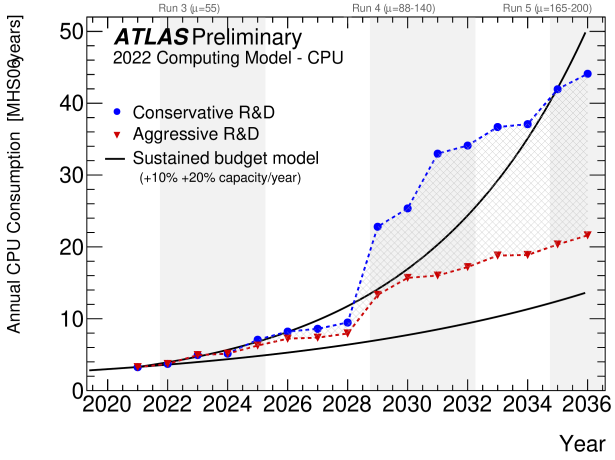
Output: local energy deposition in detector

Detector does not change within a single run
Particles do not interact with each other during transport (in detector)

Daunting evolution of required compute resources for the LHC

High-Lumi upgrade in 2028:

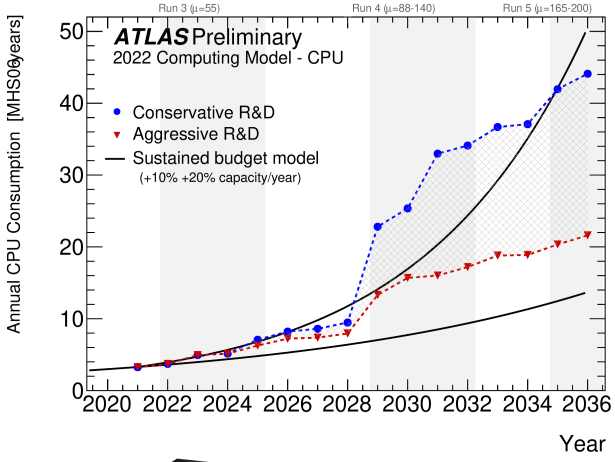
- higher statistics in simulations needed (more particles)
- simulations more expensive due to upgraded, finer detectors



Daunting evolution of required compute resources for the LHC

High-Lumi upgrade in 2028:

- higher statistics in simulations needed (more particles)
- simulations more expensive due to upgraded, finer detectors



Paradigm shift in supercomputing landscape:

17/20 of the top500 are GPU-based (NVIDIA, AMD, Intel)

20/20 of the green500 are GPU-based

➡ **GEANT4 needs to be ported to GPU to keep up with current technology**

A challenging, embarrassingly parallel problem

Huge amount of **divergence** between simulated particles due to **different geometry** and **different physics processes**

➡ first approach: only port electromagnetic part to GPU
(less complex physics, large fraction of compute time in LHC experiments)

A challenging, embarrassingly parallel problem

Huge amount of **divergence** between simulated particles due to **different geometry** and **different physics processes**

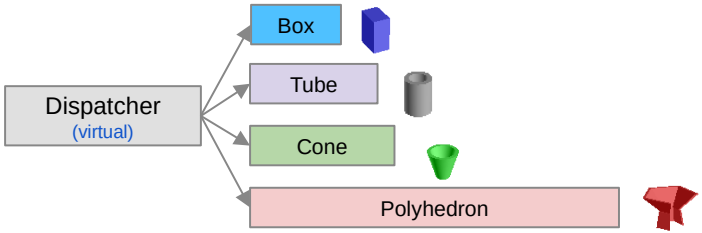
➡ first approach: only port electromagnetic part to GPU
(less complex physics, large fraction of compute time in LHC experiments)

Code structure:
AdePT (track management etc)
- G4HepEm (electromagnetic physics)
- VecGeom (geometry)

> 80% of run time on GPU spend in geometry!

Mitigating the Geometry bottleneck

Default solid model:

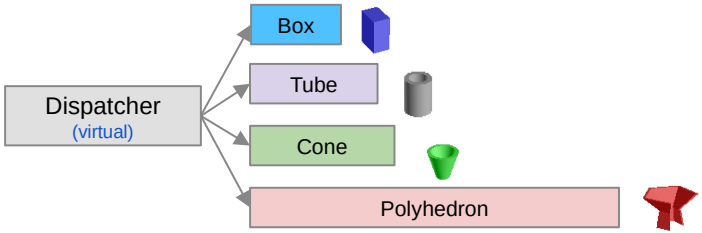


High divergence, recursive calls, virtual calls, high register usage

Not suited for GPUs!

Mitigating the Geometry bottleneck

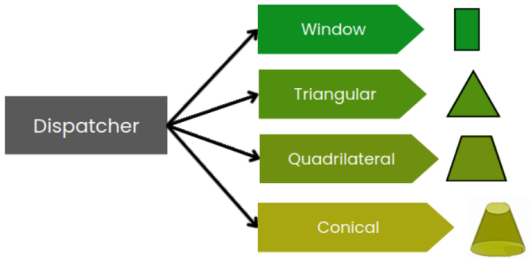
Default solid model:



High divergence, recursive calls, virtual calls, high register usage

Not suited for GPUs!

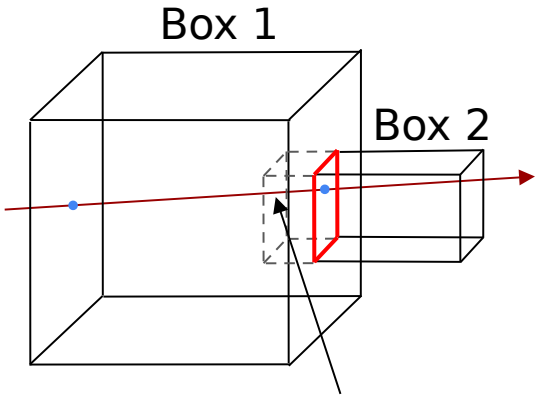
New surface model:



Reduced divergence, no recursive calls, no virtual calls, simpler code

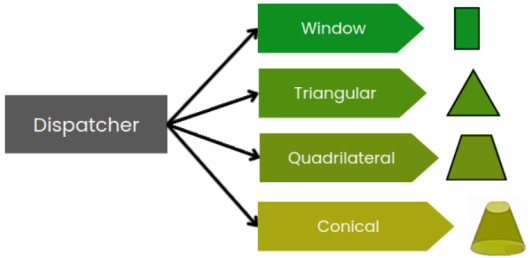
Better suited for GPUs

New model brings new challenges



Missing the overlapping entering surface leads to missing Box 2 entirely

New surface model:



Reduced divergence, no recursive calls, no virtual calls, simpler code

Better suited for GPUs

Conclusions

Despite being an embarrassingly parallel problem with millions of rays, this is an extremely challenging problem for GPUs due to code complexity, inevitable divergence, and poor memory access patterns.

Next steps:

- use of mixed precision to accelerate on GPU
- improve scheduling of tracks
- add portability:

  AdePT, VecGeom, G4HepEm ➡ mostly header-based code + some CUDA

  SYCL, Kokkos?