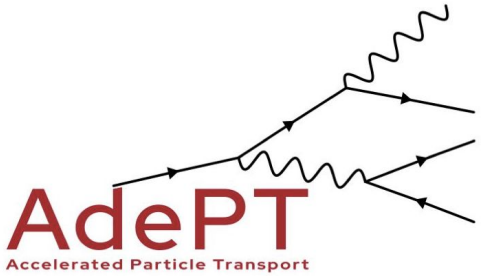


AdePT - Enabling GPU electromagnetic transport with Geant4

Juan González for the AdePT team

21/10/2024

CHEP 2024

j.gonzalez@cern.ch

Project Targets

- Understand usability of GPUs for general particle transport simulation
  - Seeking potential speed up and/or usage of available GPU resources for HEP simulation
- Provide GPU-friendly simulation components
  - Physics, geometry, field, but also data model and workflow
- Integrate in a hybrid CPU-GPU Geant4 workflow

GPU Simulation components

- Physics: G4HepEM
  - G4HepEM is a compact rewrite of EM processes, focusing on performance and targeted at HEP detector simulation applications
  - It was adapted for use on the GPU
- Geometry: VecGeom
  - GPU adaptation built on top of the original VecGeom GPU/CUDA support
  - Includes several GPU-focused improvements, like an optimised navigation state system, and a BVH navigator.
- Magnetic Field: Work in progress on a Runge Kutta field propagator
  - Currently only a uniform field with a helix propagator is validated

Recent developments

- New method for Geant4 integration
- New method for scoring
- Refactoring of AdePT into a library
  - Previously, the project consisted on a series of mostly independent examples
  - There has been a major refactoring to reorganise the project into a library, simplifying integration into external applications
  - Example integration with the GeantVal HGCAL Test Beam app
- Gaussino Integration
- Progress towards an asynchronous AdePT backend
- Major development in VecGeom’s Surface geometry model

Geant4 Integration

- Previously AdePT integrated into Geant4 applications using the Fast-simulation hooks
  - They provided an easy way to define a region for GPU transport
  - However, this approach is not flexible when trying to do GPU transport in multiple regions or even the complete geometry

- A new integration approach uses a specialized G4VTrackingManager
  - Much more customizable than the Fast-simulation hooks
  - Simplifies the integration from the user's point of view

Geant4 Integration

- AdePT Integration using the specialized AdePT Tracking Manager
  - The user only needs to register the AdePTPhysicsConstructor in their physics list
  - AdePT can be configured through an API or macro commands
  - We provided an example integration with the HGCAL Test-beam application developed by Lorenzo Pezzotti for geant-val, which can be seen in this PR
    - Besides the CMake integration of AdePT, there are minimal changes needed in the user application

Scoring

- Previously, the AdePT kernels included a simplified scoring that was done on device
  - Good for validation but not a realistic use case

- The current approach is sending back hit information, and calling the user-defined sensitive detector code on CPU
  - The sensitive volume information is taken from the geometry
  - The information coming from the GPU is used to reconstruct G4 steps
  - No changes to the user SD code are needed

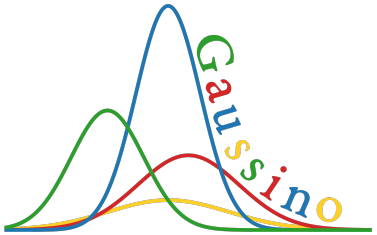
Gaussino Integration

- Gaussino is a framework allowing to configure and to steer the different phases of detector simulation
- It provides wrappers for the Geant4 physics constructors and allows to build the Geant4 modular physics list using a simple Python configuration

- Gaussino has now been extended with such a wrapper for the AdePTPhysicsPhysicsConstructor

```
GaussinoSimulation(
    PhysicsConstructors=[
        "GiGaMT_AdePTPhysics",
        "GiGaMT_G4EmStandardPhysics",
        "GiGaMT_G4EmExtraPhysics",
        "GiGaMT_G4DecayPhysics",
        "GiGaMT_G4HadronElasticPhysics",
        "GiGaMT_G4HadronPhysicsFTFP_BERT",
        "GiGaMT_G4StoppingPhysics",
        "GiGaMT_G4IonPhysics",
        "GiGaMT_G4NeutronTrackingCut"])
```

Gaussino Integration

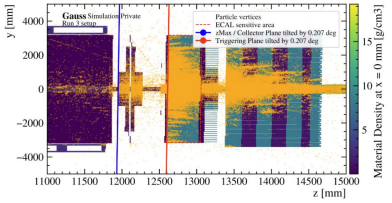


- Additional AdePT configuration can be passed through the Gaussino wrapper for Geant4 configuration macros

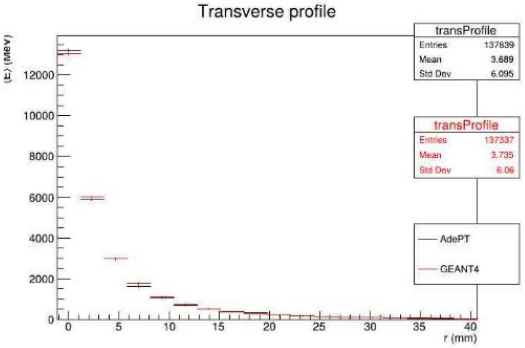
```
GiGaMTRunManagerFAC("GiGaMT.GiGaMTRunManagerFAC").InitCommands = [
   "/adept/setVecGeomGDML calochallenge.gdml",
   "/adept/addGPURegion CaloRegion" #"/adept/setTrackInAllRegions true"]
```

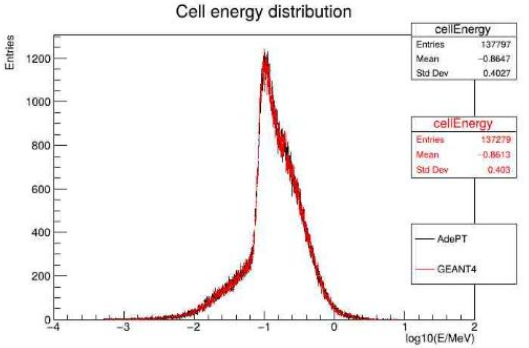
- Using the scoring mechanism discussed on slide 7
  - AdePT calls the appropriate Gaussino sensitive detectors to create hits as in a normal Geant4 simulation

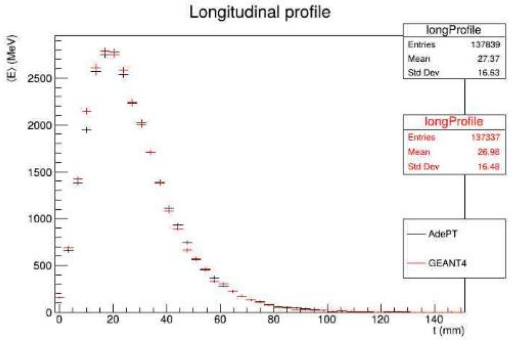
Gaussino integration - Calo Challenge setup

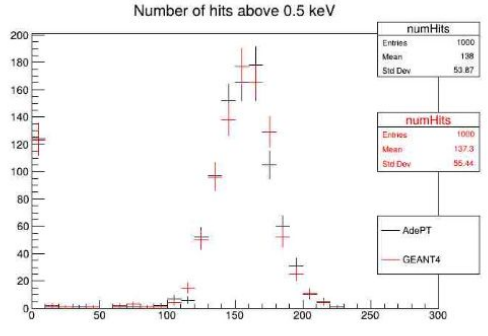


- Gaussino - AdePT integration has been successfully used for the Calo Challenge setup
- Physics results show a very good agreement with Geant4
- If there are enough particles sent to the GPU, the gains can be significant
  - Achieved 5x speedup with 4 CPU threads in initial tests with gamma-only events
  - these numbers can be considered as upper limits for real events





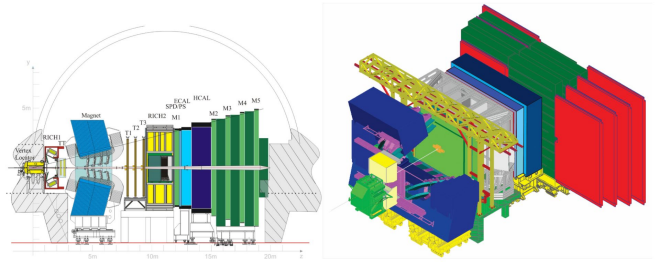




Gaussino integration - issues and next steps

- AdePT integration through Gaussino works out of the box except…
  - Full MCtruth information is not available for GPU tracks
  - It is not possible to carry over custom ‘user track information’ on the GPU
    - But custom approaches could be implemented

- Next steps
  - Currently working on full LHCb setup with AdePT integrated through Gaussino
    - Working fine out of the box, with all LHCb sensitive detectors and monitoring functioning
      - Debugging some discrepancy in the number of hits



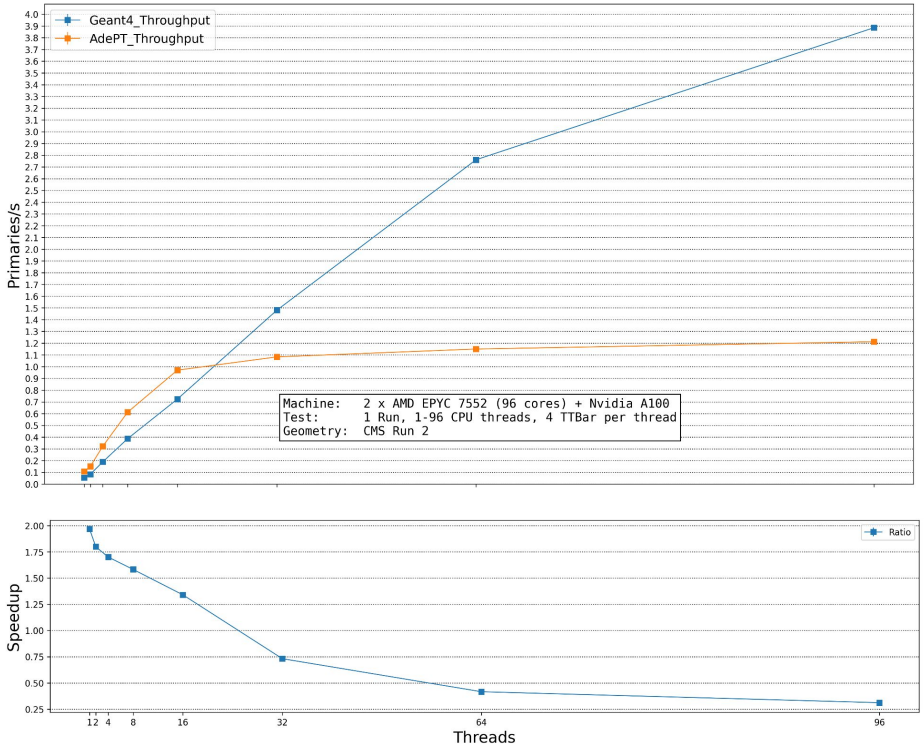
Major Issues

- We have identified two major performance bottlenecks: Geometry and kernel scheduling
  - The current solid-based geometry has two main issues on GPU:
    - The relatively large number of solid types causes warp divergence
    - The code is complex and register-hungry, which limits the maximum occupancy
  - The current approach to kernel scheduling blocks the calling thread while the GPU transports a batch of particles
    - This has a very visible effect when the GPU is saturated

Current performance results

- For low numbers of threads, offloading the EM transport to the GPU gives some speedup.
- At a certain point the GPU becomes saturated. Geometry is a major factor in how early this happens
- Due to the current way of scheduling the kernel launches, this means that the GPU starts blocking the CPU threads

```
GPU: Nvidia A100
Input: 4 TTBar per thread
Geometry: CMS2018
```

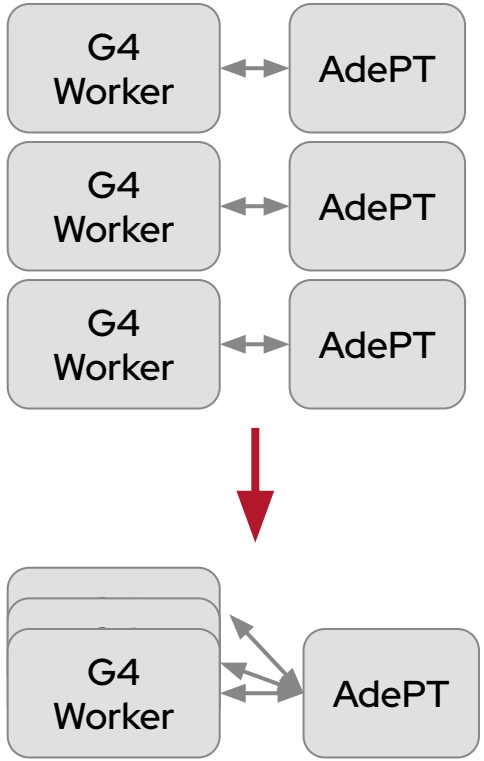


Asynchronous kernel scheduling

- Currently, CPU threads track particles across the geometry, and buffer EM particles entering marked GPU regions
- When the buffer is full, the thread triggers the transport on GPU
  - The caller blocks until the GPU finishes tracking
- As long as the GPU is not saturated this is inefficient, but we still see a speedup
- However when the GPU becomes slower, it stops the CPU from tracking particles in other regions

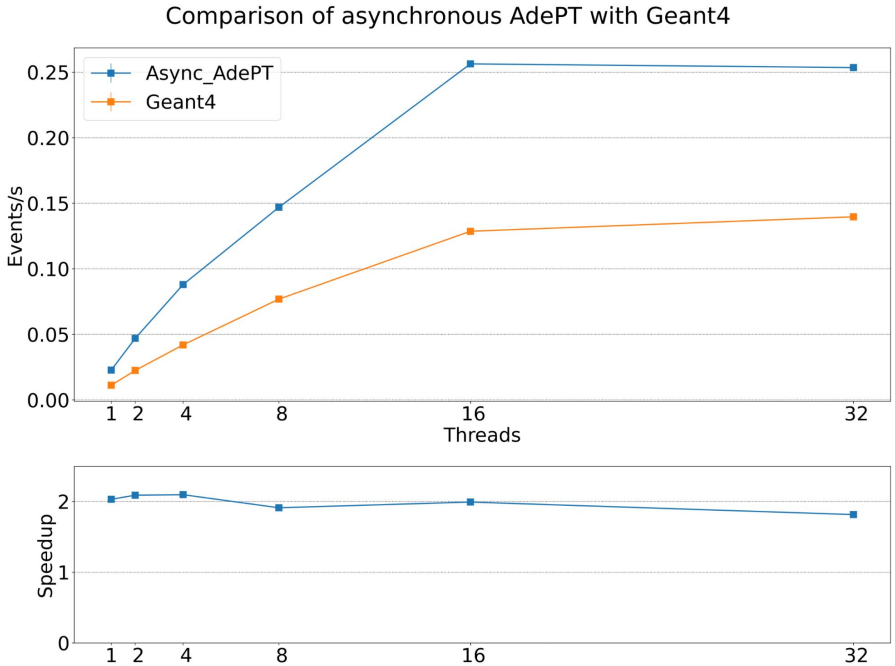
Asynchronous kernel scheduling

- Asynchronous scheduling prototype
- Only one instance of AdePT running in the background
- It continuously runs the transport loop
- All G4 workers communicate with AdePT asynchronously
    - Host threads can continue with CPU work (e.g. Hadrons) while transport runs in the background



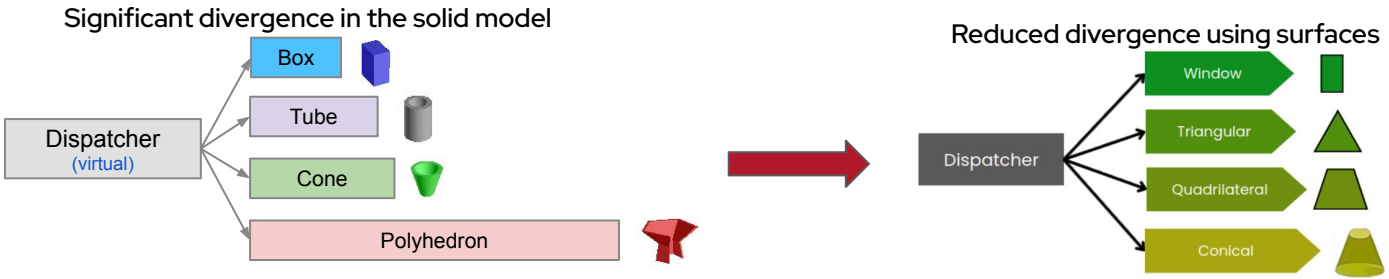
Asynchronous kernel scheduling

- Promising results in early tests
- In this example, the GPU was never saturated
- The single-threaded speedup is preserved when increasing the number of threads
- This asynchronous mode will soon be integrated into AdePT



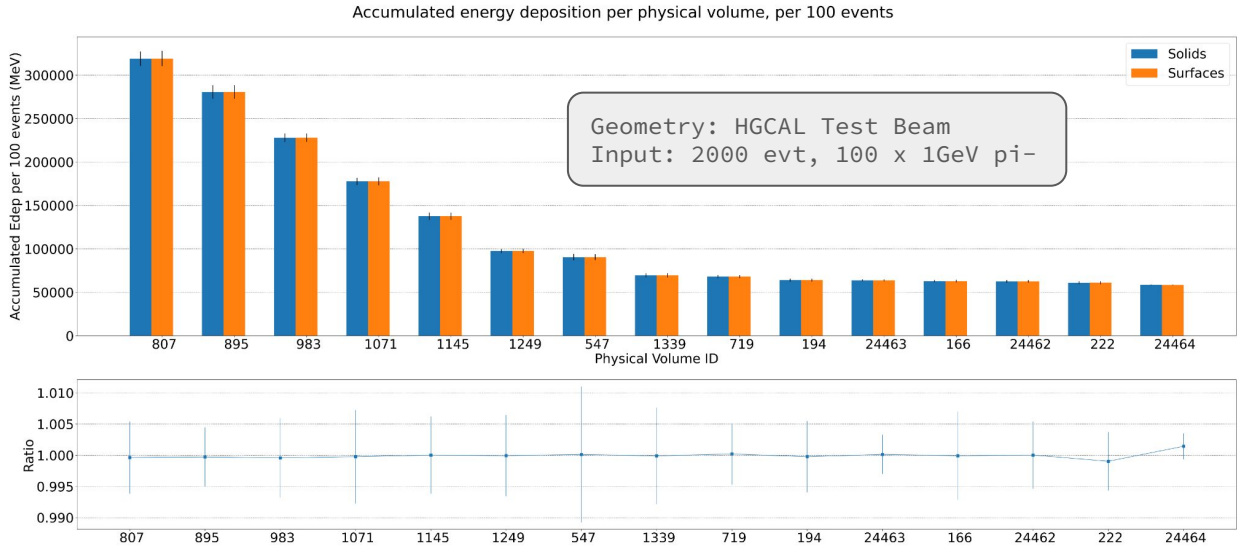
VecGeom surface model

- Simpler algorithms reduce register and stack usage
- Reduced number of primitives and lower complexity reduce divergence
- Potential to navigate using mixed precision



Status of the Surface Model

- Surface navigation already integrated into AdePT
  - Results pre-validated against solids
  - Similar performance to the solid model
    - Optimization still ongoing, with drastic performance improvements during the last months
    - Working on smaller AdePT kernels and a mixed-precision mode



Summary and outlook

- The G4TrackingManager and new approach to scoring make it easy to integrate into existing G4 applications
  - Further collaboration with experiments needed in order to find missing functionality and implement setup-specific solutions if needed
    - ATLAS AdePT/Celeritas integration hackthon on the 14th-18th October
- There are two major bottlenecks that affect the scaling behaviour
  - Poor GPU usage in the current solid-based geometry model
    - AdePT can already use the new surface model with correct results, further optimization is in progress
  - Suboptimal kernel scheduling
    - A new asynchronous mode is promising and will be implemented into AdePT in the near future