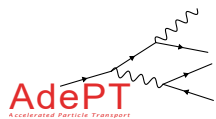
AdePT physics

HSF Detector Simulation on GPU Community Meeting

Jonas Hahnfeld & Mihály Novák

May 5, 2022



Using G4HepEm for AdePT

- Motivation: reuse code for EM processes
  - Reduce time investment for realistic demonstrator

Using G4HepEm for AdePT

- Motivation: reuse code for EM processes
  - Reduce time investment for realistic demonstrator
- Requirements:
  - Decoupling from `G4HepEmTLData`
    - Has pointer to (thread-local) random number generator
    - Stores current track as `G4HepEmElectronTrack` or `G4HepEmGammaTrack`
    - Provides temporary storage for secondaries (to be returned to GEANT4)
    - Need radically different approach on the GPU

Using G4HepEm for AdePT

- Motivation: reuse code for EM processes
  - Reduce time investment for realistic demonstrator

- Requirements:
  - Decoupling from `G4HepEmTLData`
    - Has pointer to (thread-local) random number generator
    - Stores current track as `G4HepEmElectronTrack` or `G4HepEmGammaTrack`
    - Provides temporary storage for secondaries (to be returned to GEANT4)
    - Need radically different approach on the GPU

  - Abstraction of random number generator
    - CLHEP on the CPU when used with GEANT4 (default: MIXMAX)
    - Want to use RANLUX++ on the GPU
      (smaller state, but equally excellent statistical properties)

Decoupling from `G4HepEmTLData`

- Two main functions in G4HepEm: `HowFar` and `Perform`
- Accept pointer to `G4HepEmTLData`, stored in GEANT4 process

Decoupling from `G4HepEmTLData`

- Two main functions in G4HepEm: `HowFar` and `Perform`
- Accept pointer to `G4HepEmTLData`, stored in GEANT4 process

- On the GPU,
  - Tracks are stored in arrays (by type), and have their own RNG state
  - `G4HepEmElectronTrack` or `G4HepEmGammaTrack` are constructed temporarily

Decoupling from G4HepEmTLData

- Two main functions in G4HepEm: `HowFar` and `Perform`
- Accept pointer to `G4HepEmTLData`, stored in GEANT4 process

- On the GPU,
  - Tracks are stored in arrays (by type), and have their own RNG state
  - `G4HepEmElectronTrack` or `G4HepEmGammaTrack` are constructed temporarily

- Solution: rework functions for reuse on GPU
  - Split functions into smaller pieces (example: continuous energy loss)
  - Overload functions to also accept explicit pointer to track and RNG (if needed)
- Implemented in a series of PRs: `mnovak42/g4hepem#16`, `mnovak42/g4hepem#21`, `mnovak42/g4hepem#25`, `mnovak42/g4hepem#30`

Abstraction of random number generator

- Various functions need access to random numbers
  - MSC step limit randomization, scattering, and displacement
  - Final state sampling for discrete processes (energy transfer & directions)

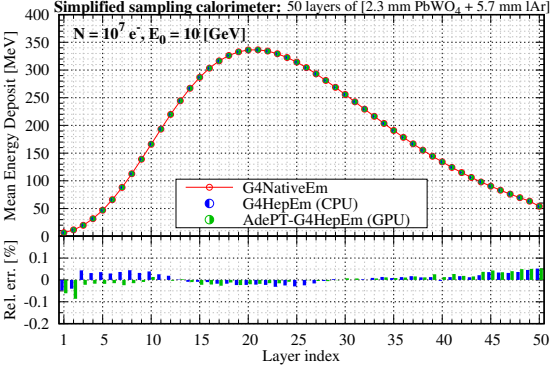
Abstraction of random number generator

- Various functions need access to random numbers
  - MSC step limit randomization, scattering, and displacement
  - Final state sampling for discrete processes (energy transfer & directions)

- Tiny wrapper class `G4HepEmRandomEngine` (`mnovak42/g4hepem#18`)
  - Reference to real engine and two function pointers
- Relevant selection and sampling functions take an argument of this class (instead of `G4HepEmTLData`, see previous slide)

Validation: TestEm3, $PbWO_4$ + lAr, 10 GeV electrons

GEANT4 11.0, complete physics including MSC except energy loss fluctuations



Mean values per event of some quantities:

	GEANT4	G4HepEm			
	G4Em (CPU)	HepEm (CPU)	**Rel. err.** [%]	AdePT (GPU)	**Rel. err.** [%]
Energy deposit per material [MeV]					
$PbWO_4$	6730.00	6729.46	-0.008	6729.46	-0.008
lAr	2566.18	2566.55	0.014	2566.52	0.013
Number of secondaries					
γ	4456.06	4454.57	-0.033	4451.46	-0.103
e^-	8066.48	7953.90	-1.40*	7953.22	-1.40*
e^+	429.103	429.146	0.010	429.147	0.010
Number of steps					
charged	36696.7	36283.2	-1.13*	36292.0	-1.10*
neutral	40377.9	40426.6	0.121	40597.9	0.545

Energy loss fluctuations on GPU

- Implemented in G4HepEm; possible to run on GPU, but not enabled in AdePT

Energy loss fluctuations on GPU

- Implemented in G4HepEm; possible to run on GPU, but not enabled in AdePT
- TestEm3 with Pb/lAr, simulating 100k electrons, 10 GeV (RTX 2070 SUPER)
  - Current `master`: 183s
  - With fluctuations: 260s; + 42 %
    → significant slowdown compared to around 10 % on the CPU

Energy loss fluctuations on GPU

- Implemented in G4HepEm; possible to run on GPU, but not enabled in AdePT
- TestEm3 with Pb/lAr, simulating 100k electrons, 10 GeV (RTX 2070 SUPER)
  - Current `master`: 183s
  - With fluctuations: 260s; + 42 %
    → significant slowdown compared to around 10 % on the CPU
- Influence smaller for more complex geometries
  - Around + 10 % for `cms2018`, compared to 2-3 % on the CPU

Energy loss fluctuations on GPU (contd.)

Especially problematic: loop for sampling energy loss due to ionization

Energy loss fluctuations on GPU (contd.)

Especially problematic: loop for sampling energy loss due to ionization

1. High thread divergence:
   - Number of loop iterations determined by drawing from Poisson distribution

Energy loss fluctuations on GPU (contd.)

Especially problematic: loop for sampling energy loss due to ionization

1. High thread divergence:
   - Number of loop iterations determined by drawing from Poisson distribution
2. Many random numbers needed
   - Each loop iteration needs one random number
   - Possibility: Use "weaker" / faster RNG for this loop
     $\rightarrow$ Using xorshift* reduces slowdown to "only" 25 % for TestEm3

Energy loss fluctuations on GPU (contd.)

Especially problematic: loop for sampling energy loss due to ionization

1. High thread divergence:
   - Number of loop iterations determined by drawing from Poisson distribution
2. Many random numbers needed
   - Each loop iteration needs one random number
   - Possibility: Use "weaker" / faster RNG for this loop
     $\rightarrow$ Using xorshift* reduces slowdown to "only" 25 % for TestEm3

Current conclusion: fluctuations not suitable for simulation on GPU - needed for HEP?

Conclusion

- Possible to run full EM shower simulation on GPUs!

Conclusion

- Possible to run full EM shower simulation on GPUs!
- Reusing $> 95\,\%$ of the code from G4HepEm
  - After decoupling from `G4HepEmTLData` and abstracting the RNG

Conclusion

- Possible to run full EM shower simulation on GPUs!
- Reusing $> 95\,\%$ of the code from G4HepEm
  - After decoupling from `G4HepEmTLData` and abstracting the RNG
- Excellent agreement with GEANT4
  - For the setups and energy ranges required in HEP detectors (see previous talk)