

# Integrating AdePT into Geant4 workflow

Andrei Gheata

Witek Pokorski

16.09.2021

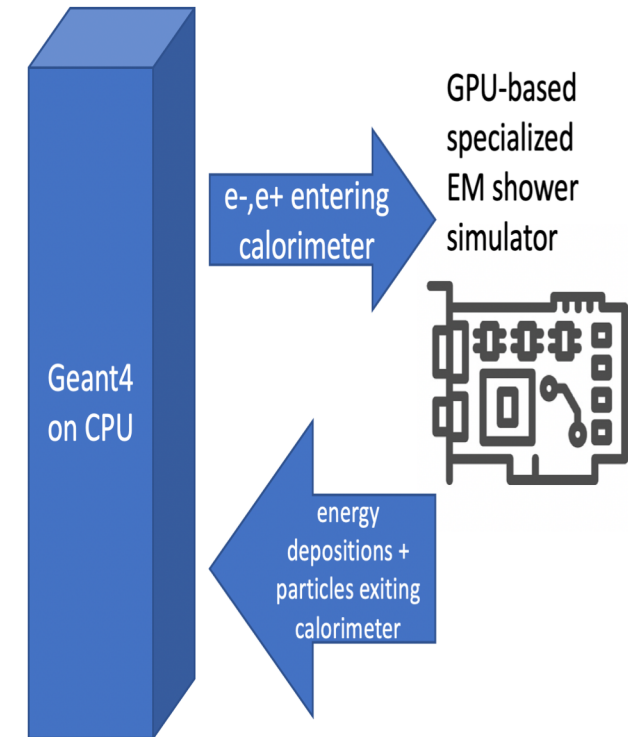
26th Geant4 Collaboration Meeting

# Introduction

- significant fraction of the the whole work needs to be performed in one go on the device due to the high cost of transferring data between CPU and GPU
  - AdePT is targeting the simulation of the entire calorimeter shower on the GPU
  - using GPU for any 'smaller bits' of the whole simulation (like specific processes, or tracks) is very unlikely to bring any benefits (or rather the opposite)
- the communication (Geant4 <> external library) should be reduced to minimum and grouped as much as possible
  - minimizing synchronization constraints
- we want to profit from massive parallelism by transforming the calorimeter shower simulation, in a normal Geant4 CPU workflow, in a "hotspot" handled by the GPU

# Prototype integration strategy

- region-based approach for delegating simulation to an external transport
  - particle killed on the Geant4 side and passed to the other transport engine
  - energy depositions and 'outgoing' (from that region) particles returned
- this follows 'fast-simulation' approach in Geant4
  - allows to use (most of the) existing fast-simulation hooks
  - easy integration with the physics list
  - ability to switch between full Geant4 and Geant4 + 'fast sim' at runtime (from macro file)
- one difference:
  - we want to be able to group particles to process them together



# AdePT as fast simulation process (1/2)

- uses all the existing infrastructure of Fast Simulation Physics
- registered and activated in the standard way

```
FTFP_BERT* physicsList = new FTFP_BERT; // G4VModularPhysicsList
auto fastSimulationPhysics = new G4FastSimulationPhysics(); // helper
fastSimulationPhysics->ActivateFastSimulation("e-"); // for envelope in mass geometry
fastSimulationPhysics->ActivateFastSimulation("pi+", "pionGhostWorld"); // for envelopes in para
physicsList->RegisterPhysics( fastSimulationPhysics ); // attach to the physics list
```

# AdePT as fast simulation process (2/2)

- interfacing done in the usual way through G4VFastSimulationModel interface
  - class AdePTModel : public G4VFastSimulationModel implements following methods
    - **IsApplicable, ModelTrigger** – standard functionality
    - **Initialize** – instantiates AdePT engine per thread
    - **Dolt** – passes particles from Geant4 to AdePT by calling fAdept->AddTrack method
      - see next slide
    - **Flush** – ‘flushes’ the AdePT buffer before the end of the Geant4 event
      - see following slide

# Dolt method

- calls AddTrack(...) method of AdePT
- AdePT puts the track on the buffer and checks its size with respect some (adjustable) threshold
  - if size below the threshold, just add to the buffer and go back
  - if threshold is reached
    - AdePT::Shower method is called
      1. calls AdePT::ShowerGPU method (CUDA)
        - transports the contents of the buffer through the calorimeter geometry on GPU
      2. loops through the buffer of produced particles (outgoing from the calorimeter), creates corresponding Geant4 tracks and puts them on the Geant4 stack
      3. loops through the produced energy depositions per volume and converts them to Geant4 hits

```
void AdePTModel::Dolt(const G4FastTrack, G4FastStep)
{
    ...
    fAdept->AddTrack(pdg, energy, particlePosition, particleDirection);
}
```

```
void AdePT::AddTrack(...)
{
    fBuffer.toDevice.emplace_back(...);

    if(fBuffer.toDevice.size() >= kBufferThreshold) Shower();
}
```

```
void AdePT::Shower(int event)
{
    // run the shower simulation on GPU
    AdePT::ShowerGPU(event, fBuffer);

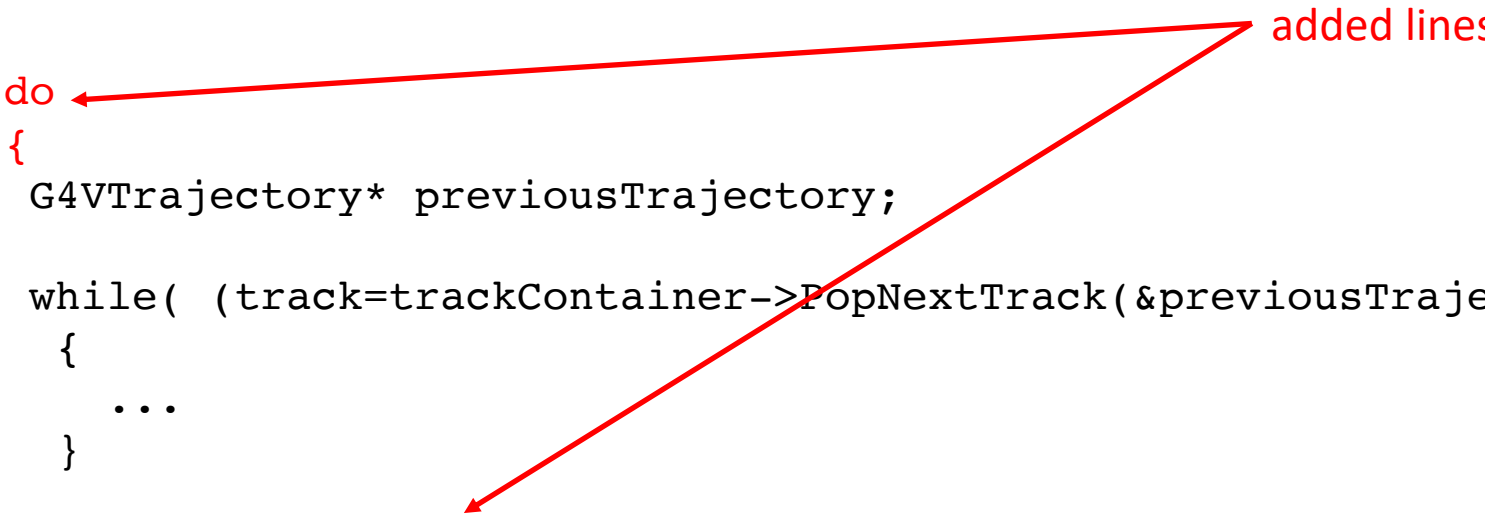
    // get secondaries and put them on Geant4 stack
    for(auto track:fBuffer.fromDevice) {...}

    // get energy deposition per sensitive volume and create G4 hits
    ...
}
```

# Flush method

- new with respect to the existing implementation
  - maybe not the best name, please propose better
- as discussed on previous slide, simulation of particles on GPU is triggered by the buffer occupancy reaching some threshold
  - but when Geant4 stack is empty, we need to run the GPU simulation for remaining particles in AdePT buffer (below threshold) before closing the event
    - AdePT simulation may produce secondaries that will be put back on the Geant4 stack
  - none of the existing (user) methods can play this role
- new call needed in Event Manager
  - happening when there are no more particles on Geant4 stack, but before event is finished
  - need to dispatch the call to all fast simulation models
    - done through G4GlobalFastSimulationManager

# G4EventManager.cc



```
do
{
    G4VTrajectory* previousTrajectory;

    while( (track=trackContainer->PopNextTrack(&previousTrajectory)) != nullptr )
    {
        ...
    }

    G4GlobalFastSimulationManager::GetGlobalFastSimulationManager()->Flush();

} while (trackContainer->GetNUrgentTrack() > 0);
```

added lines



# G4GlobalFastSimulation Manager and G4FastSimulationManger

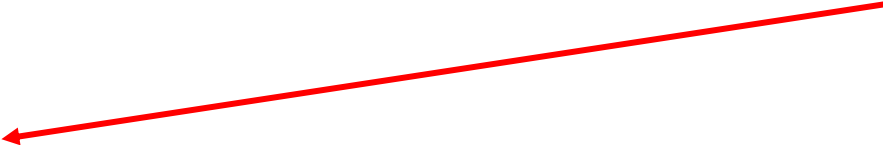
```
void G4GlobalFastSimulationManager::Flush()  
{  
    // loop over all model managers and flush  
  
    for (size_t ifsm=0; ifsm<ManagedManagers.size(); ifsm++)  
        ManagedManagers[ifsm]->FlushModels();  
}
```

```
void G4FastSimulationManager::FlushModels()  
{  
    for (size_t iModel=0; iModel<ModelList.size(); iModel++)  
        ModelList[iModel]->Flush();  
}
```

# G4VFastSimulationModel

```
class G4VFastSimulationModel
{
...
public:
    virtual void Flush(){}
}
```


added this virtual function  
(empty by default)



## concrete implementation:

```
void AdePTModel::Flush()
{
    fAdept->Shower();
}
```

triggers the shower  
(shortcutting the Dolt method)  
on what is left in the buffer  
(below threshold)



# Conclusion

- for GPU-based transport to be beneficial, simulation of entire detector regions need to be delegated to it
- this is very similar to the idea ‘fast simulation’ plugins in Geant4
- existing fast simulation hooks provide ‘ready to use’ functionality
  - allow very nice integration and ability to switch AdePT ‘module’ on/off at run time
- one missing element is the call to the ‘flush buffer’ method before the end of even
  - proposed prototype implementation provides it with full backwards compatibility and minimal overhead
- as discussed on [Tuesday](#), integration of G4HepEm in Geant4 requires different, ‘per particle’ approach (maybe using specialized stacks)
  - most likely, we need the combination of the two