



AdePT

Offloading electromagnetic showers in Geant4 simulations to GPU

Severin Diederichs, on behalf of the AdePT team

severin.diederichs@cern.ch

CERN, EP-SFT

WLCG/HSF Workshop 2025, IJCLab, Paris

Full Simulation must become more efficient

Last presentation:

~ 20% performance improvement in full production simulations for ATLAS and CMS using specialized tracking for e^- , e^+ , and γ on CPU with **G4HepEm**

Full Simulation must become more efficient ... using GPUs?

Last presentation:

~ 20% performance improvement in full production simulations for ATLAS and CMS using specialized tracking for e^- , e^+ , and γ on CPU with **G4HepEm**

This presentation:

~ X% performance improvement in full production simulations for ATLAS and CMS using specialized tracking for e^- , e^+ , and γ on GPU with **AdePT**

Full Simulation must become more efficient ... using GPUs?

Last presentation:

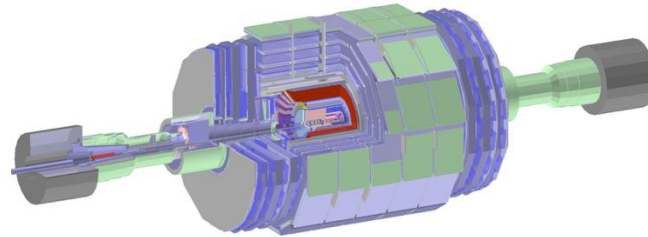
~ 20% performance improvement in full production simulations for ATLAS and CMS using specialized tracking for e^- , e^+ , and γ on CPU with **G4HepEm**

This presentation:

Not there yet! Current status of reasonably realistic simulations
~~X% performance improvement in full production simulations for ATLAS and CMS~~
using specialized tracking for e^- , e^+ , and γ on GPU with **AdePT**

Full Simulation on GPU is a hard problem

- Random memory access
- Divergence in geometry
- Divergence in physics



vs



AdePT core components

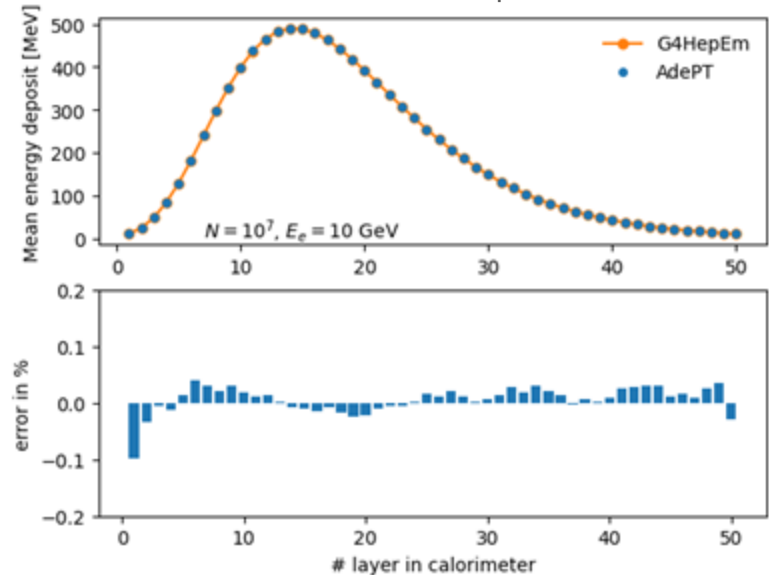
- TrackingManager, kernel scheduling, and transport
- Physics: **G4HepEm**
 - Provides physics functions on GPU and specialized tracking on CPU
- Geometry: **VecGeom** with GPU backend
 - Using an in-memory converter from G4 solids to VecGeom solid / surface model
- Magnetic field: Dormand-Prince integrator
 - Using the **covfie** library for 3D field maps



Physics correctness

AdePT relies fully on **G4HepEm physics*** in custom tracking loop including $e^{-/+}$ -and γ -nuclear processes

Longitudinal energy profile in simple calorimeter with nuclear processes



* does not mean that we get the same physics results due to custom tracking loop, VecGeom, and B field propagation

Physics correctness

AdePT relies fully on **G4HepEm physics*** in custom tracking loop including $e^{-/+}$ -and γ -nuclear processes

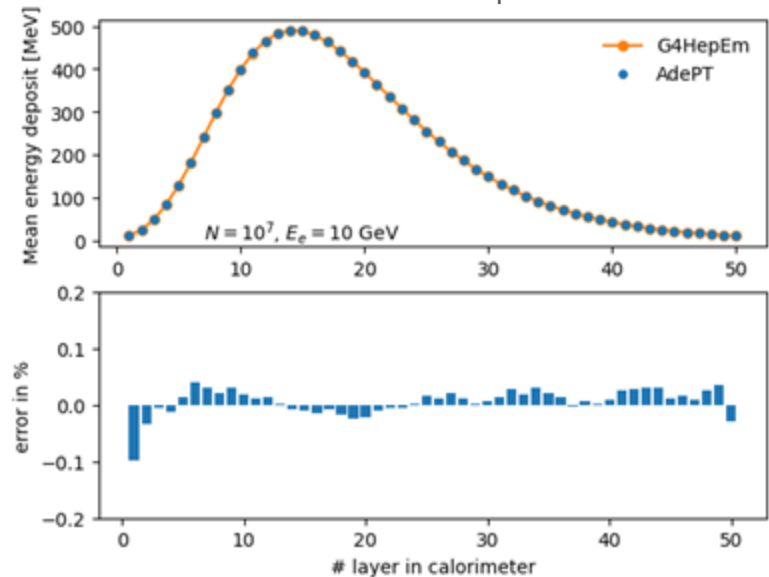
Missing optimizations:

- Woodcock tracking
- Multiple steps in MSC with transport

Fallback to CPU via the *G4HepEmTrackingManager*:

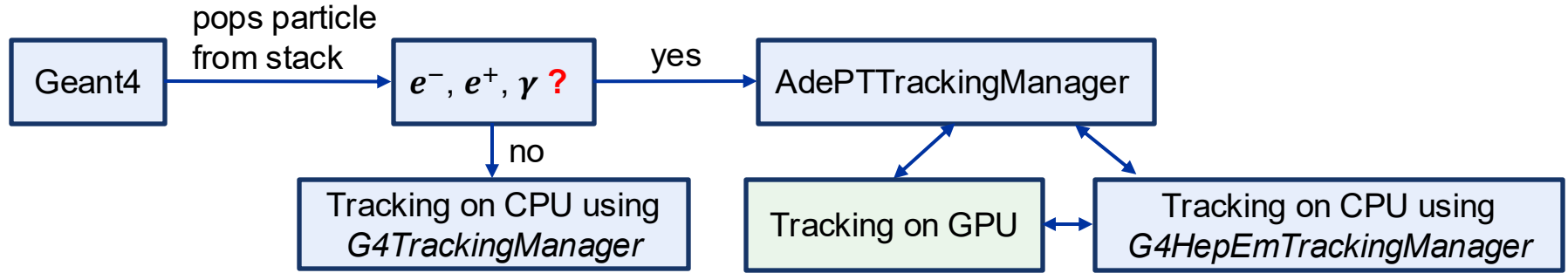
- Nuclear processes
- Partial usage of FastSim
- Custom processes in certain regions (e.g. transition radiation)

Longitudinal energy profile in simple calorimeter with nuclear processes

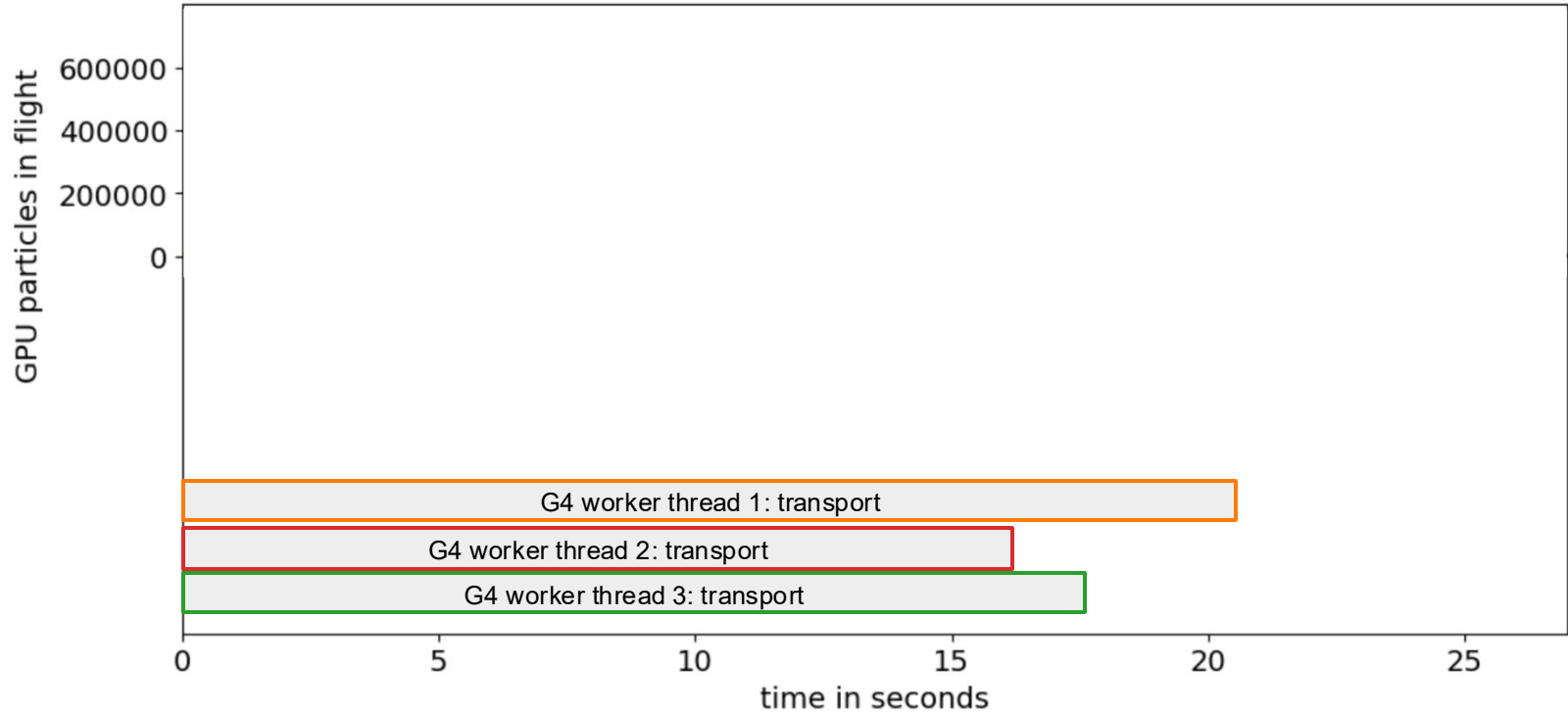


* does not mean that we get the same physics results due to custom tracking loop, VecGeom, and B field propagation

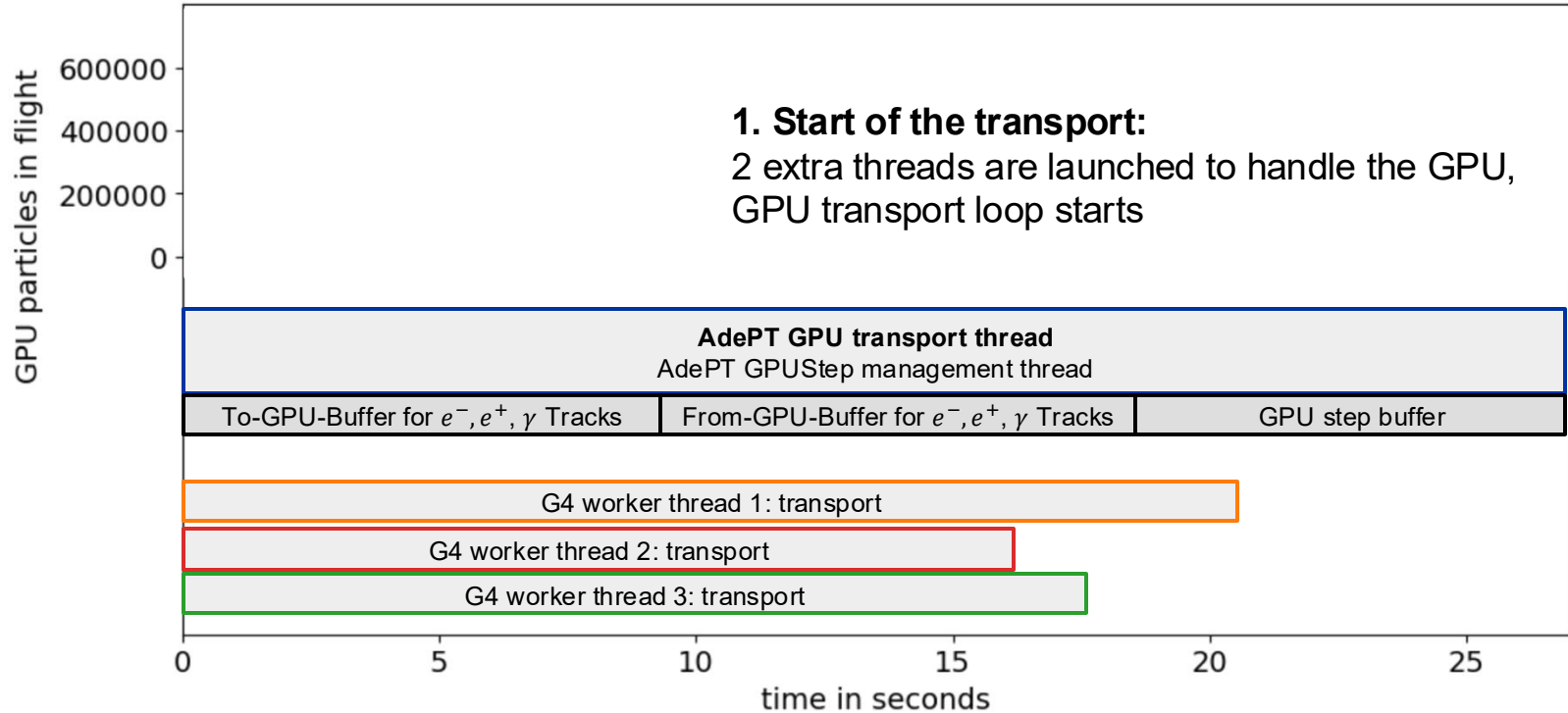
Offloading made easy via the *G4VTrackingManager*



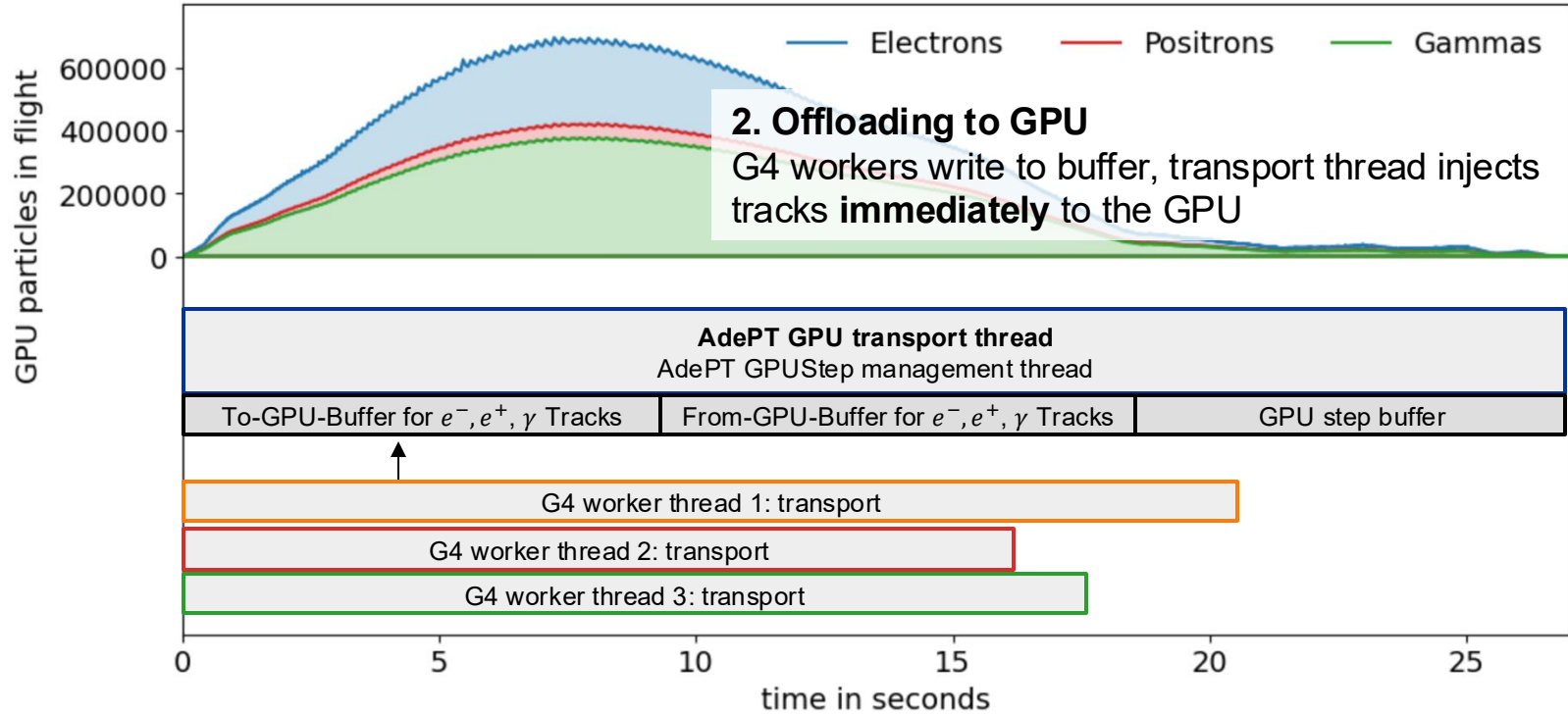
Asynchronous GPU-offloading in AdePT



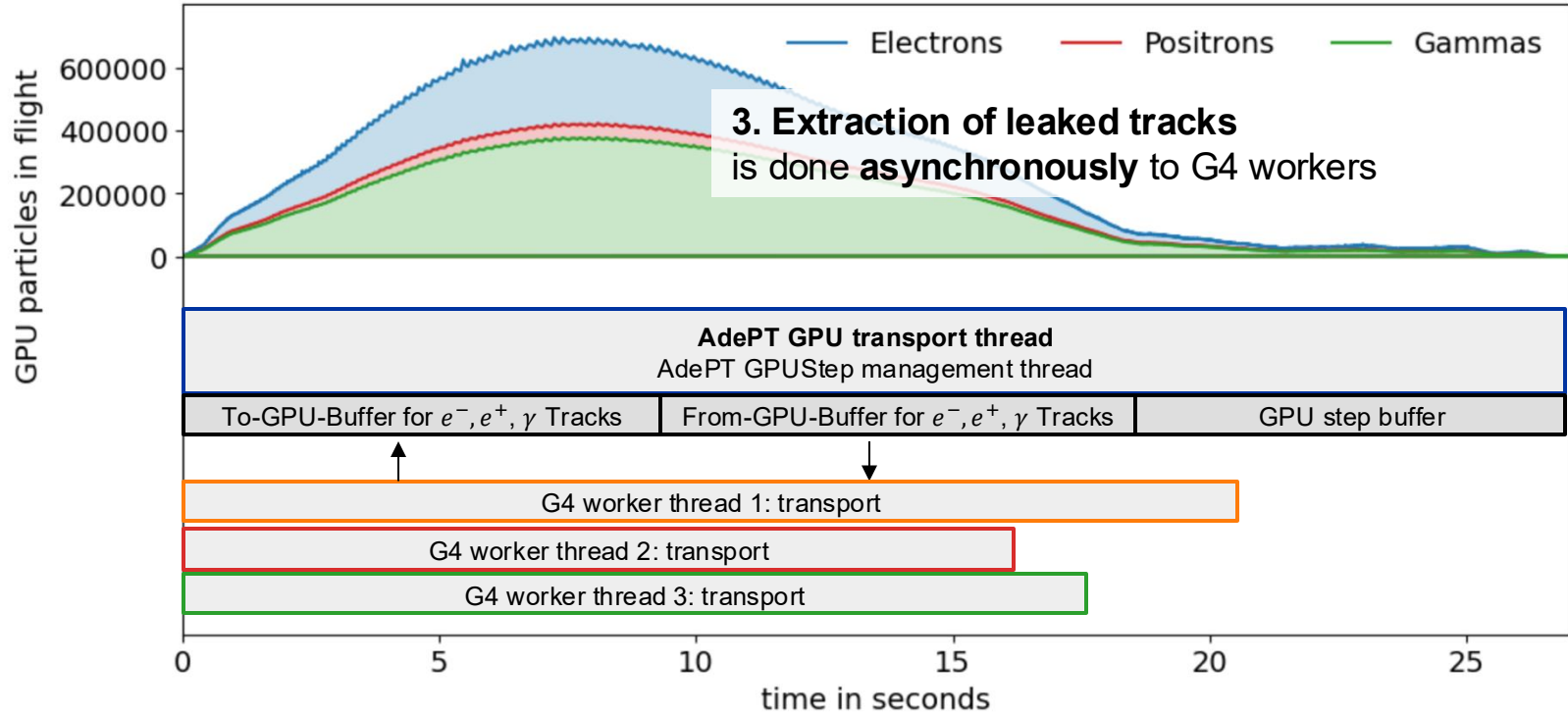
Asynchronous GPU-offloading in AdePT



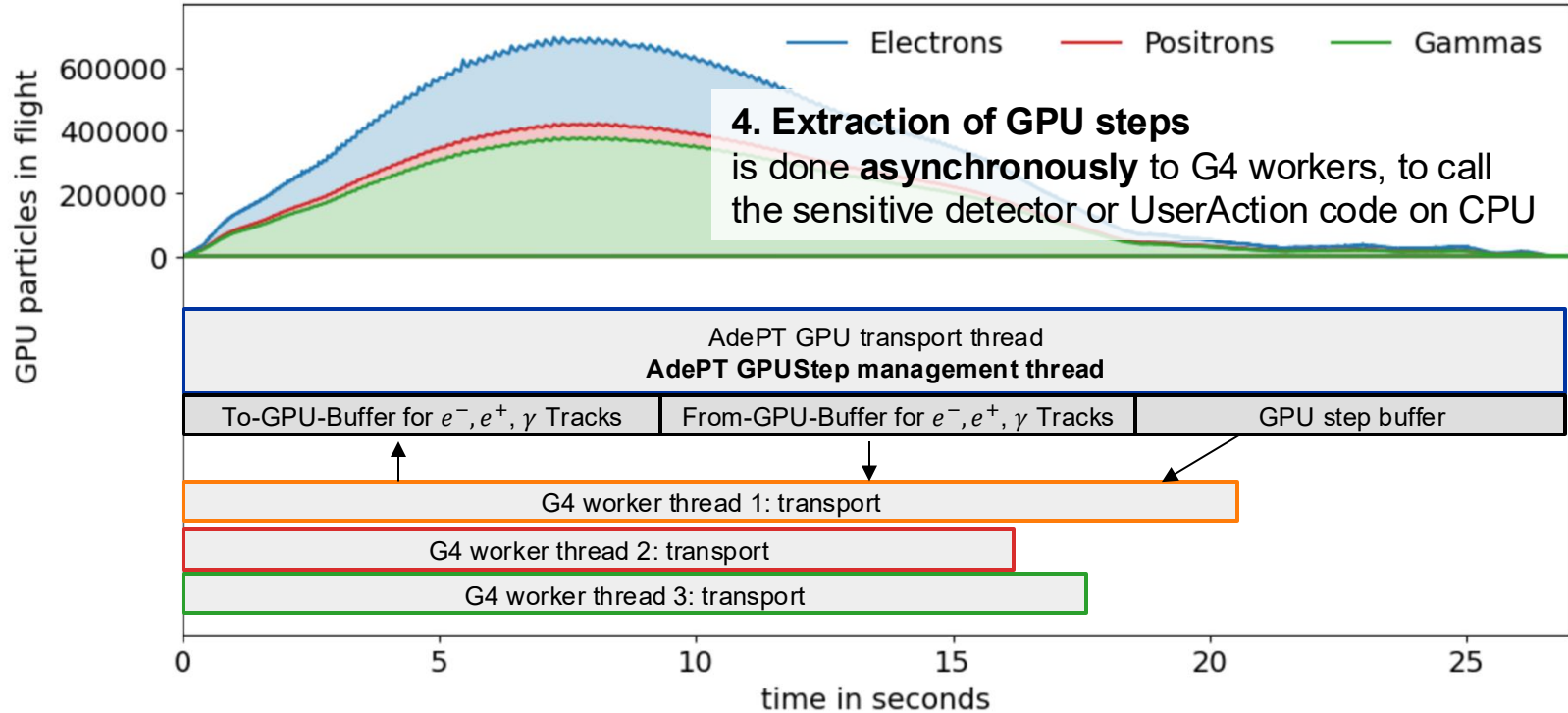
Asynchronous GPU-offloading in AdePT



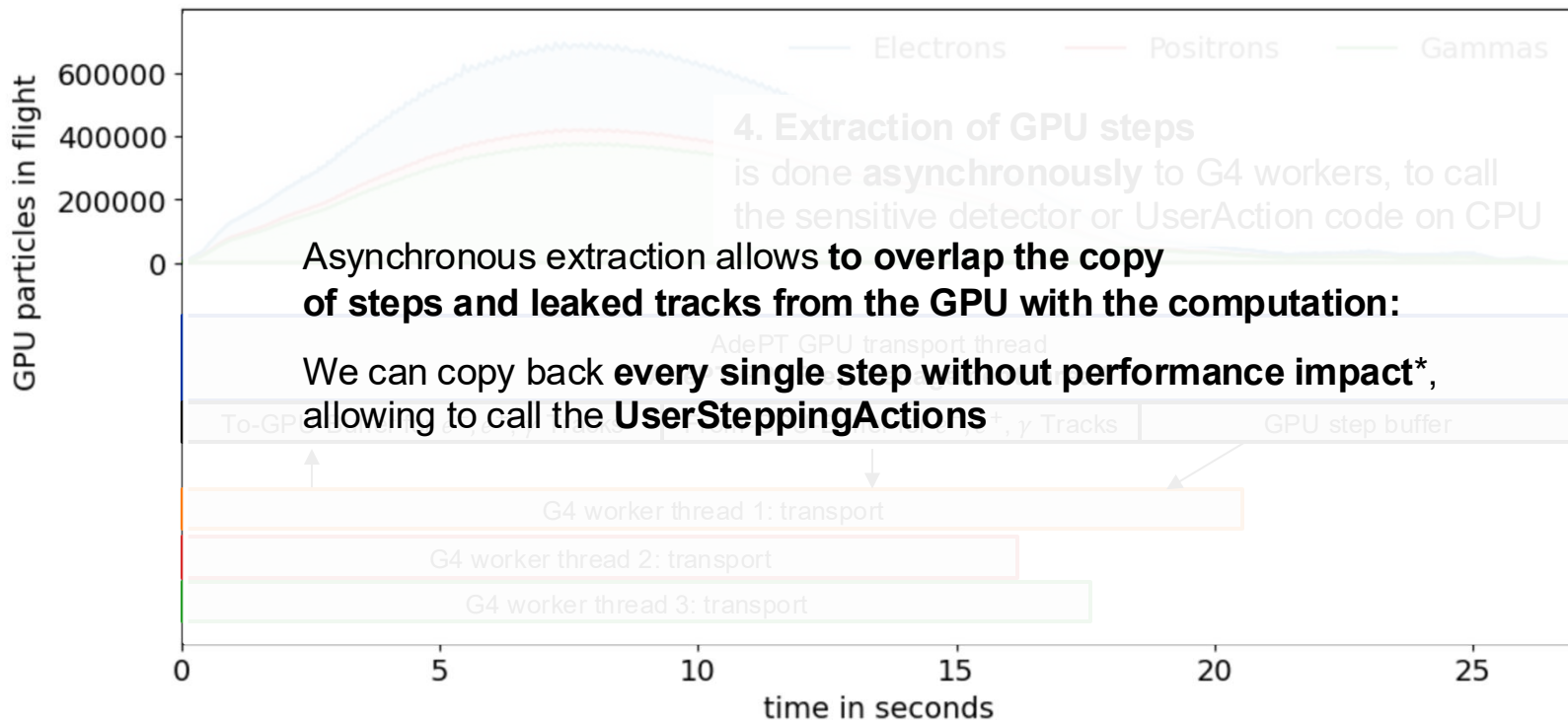
Asynchronous GPU-offloading in AdePT



Asynchronous GPU-offloading in AdePT



Asynchronous GPU-offloading in AdePT

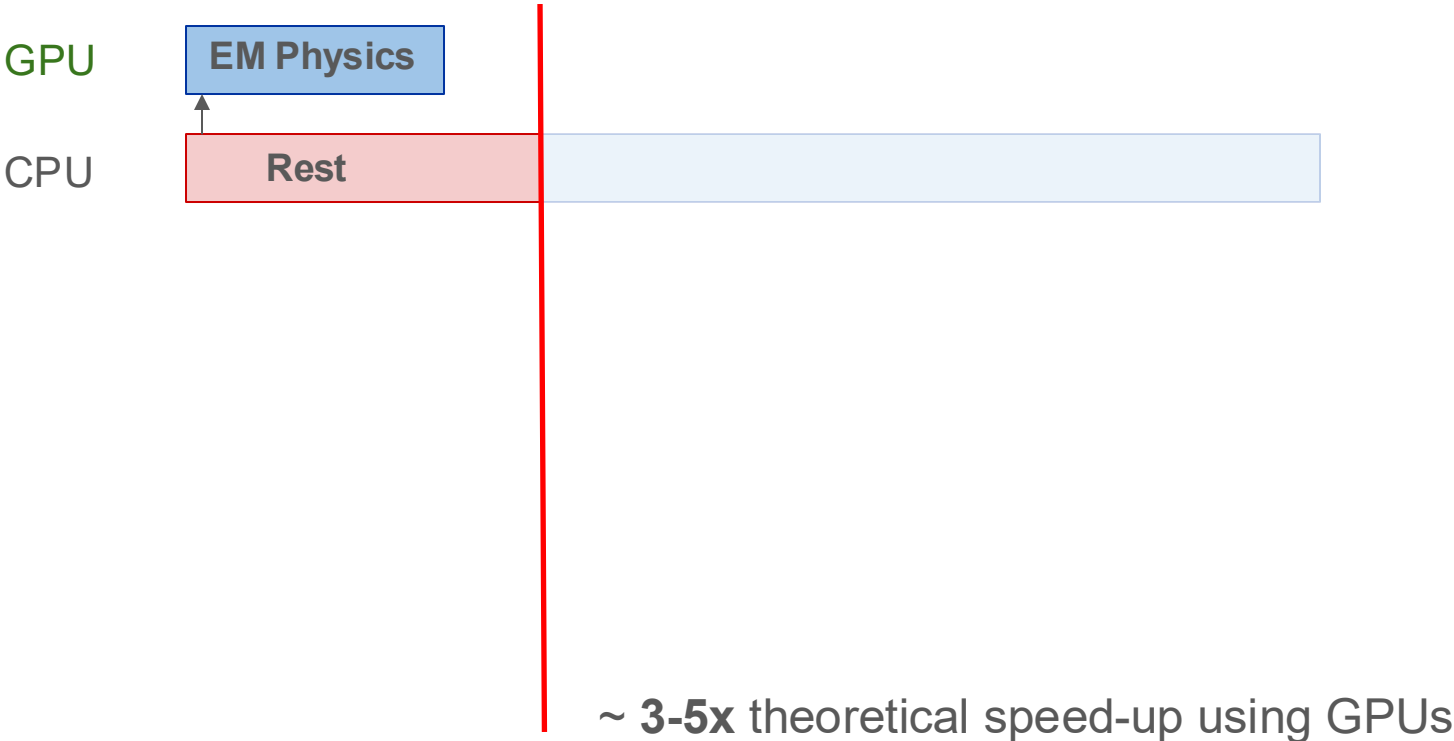


Understanding the target: port LHC production runs to GPU

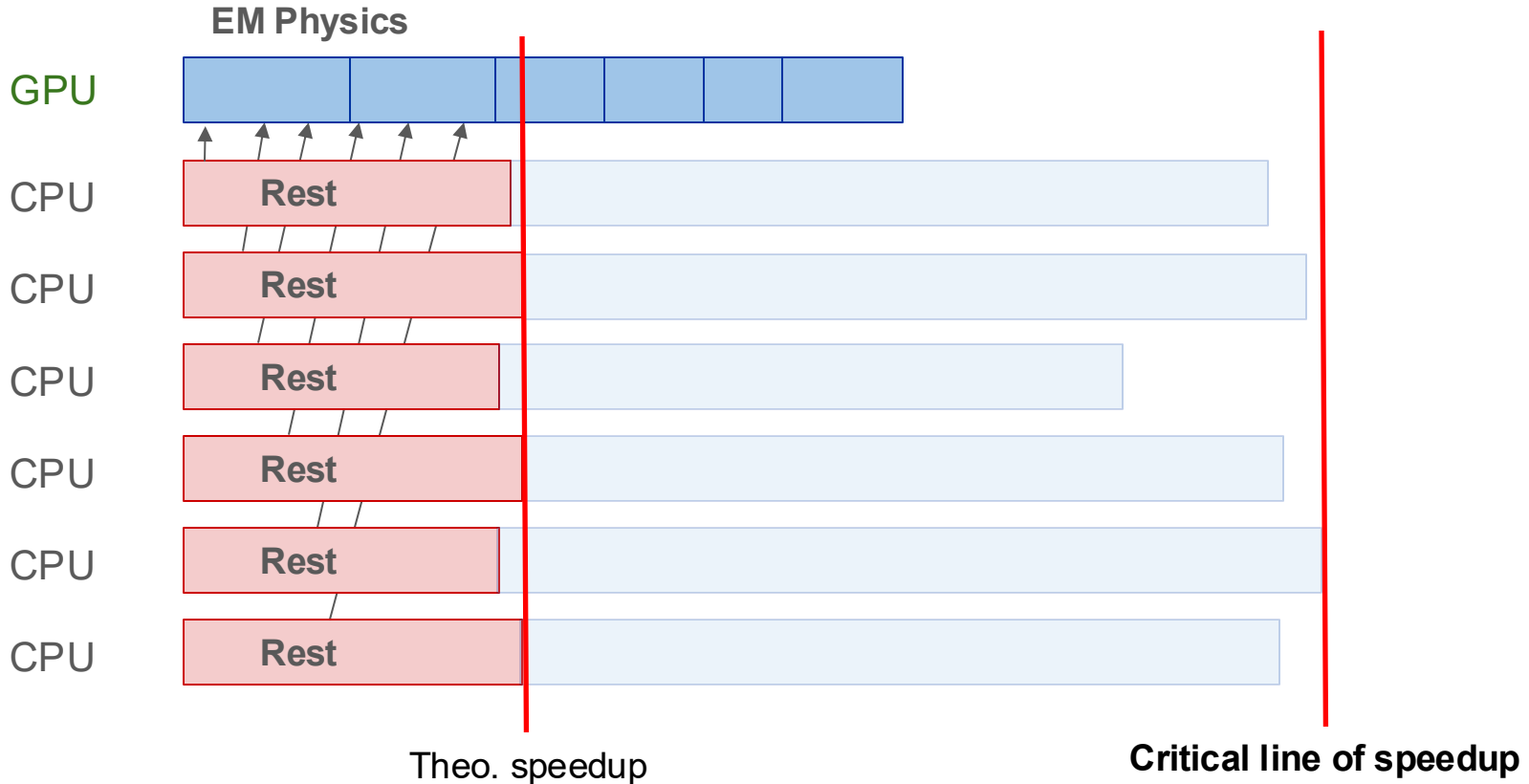
CPU



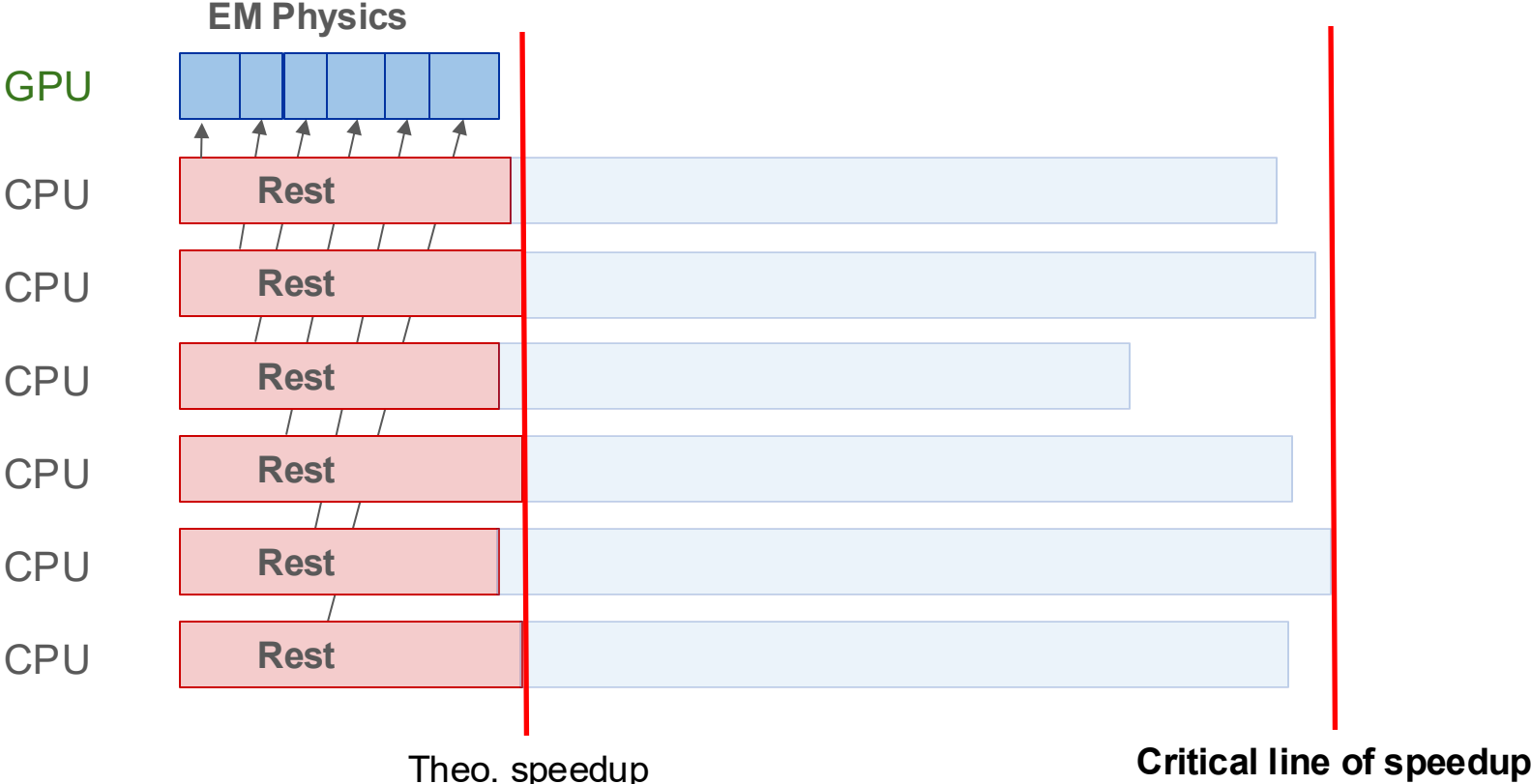
Understanding the target: port LHC production runs to GPU



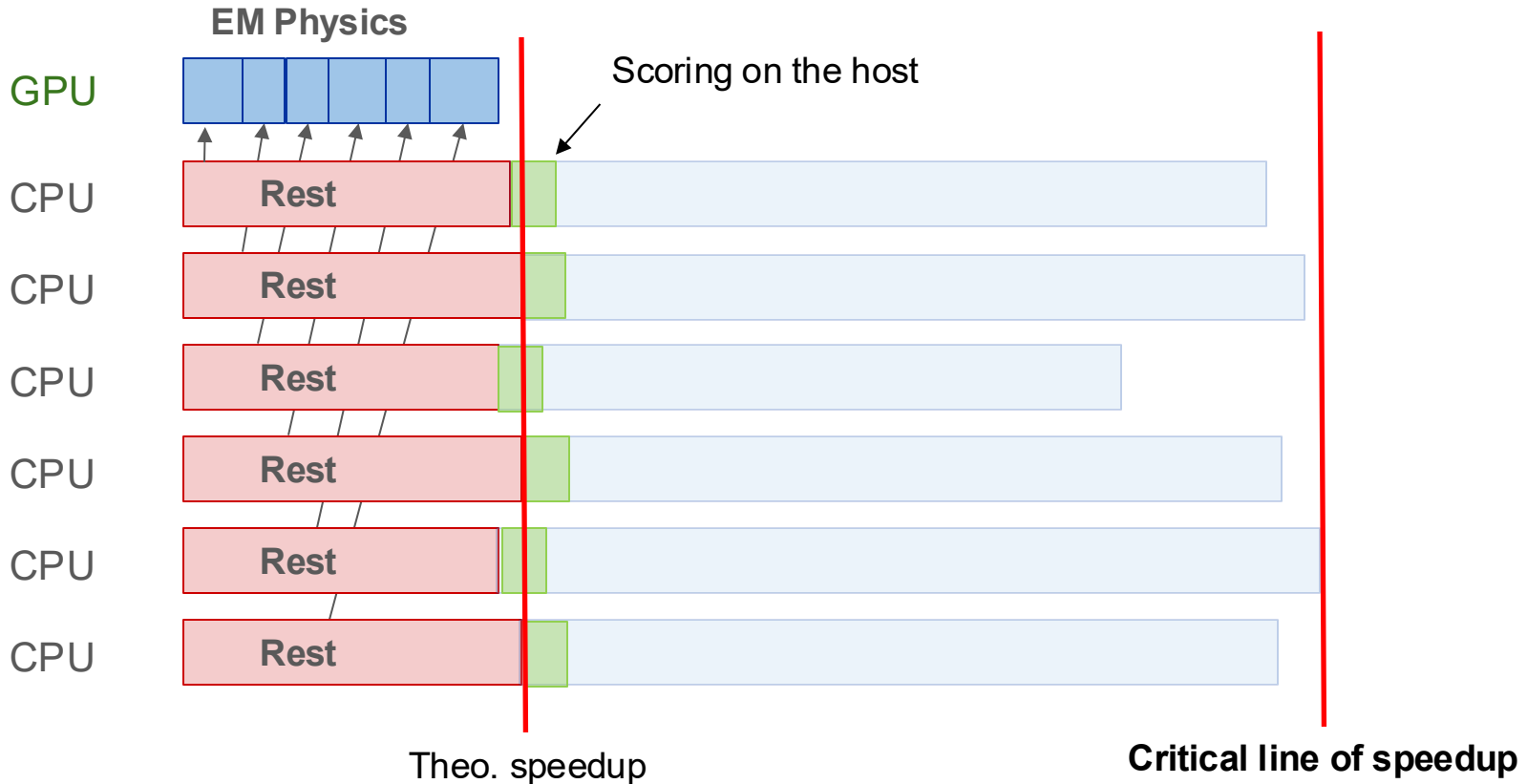
To achieve speedup, the GPU must be *fast*



Theo. speedup: complete overlap of GPU and CPU computation

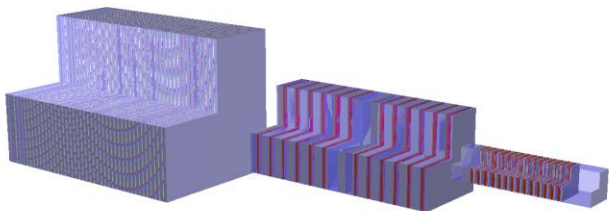


Reality: not perfectly achievable due to scoring

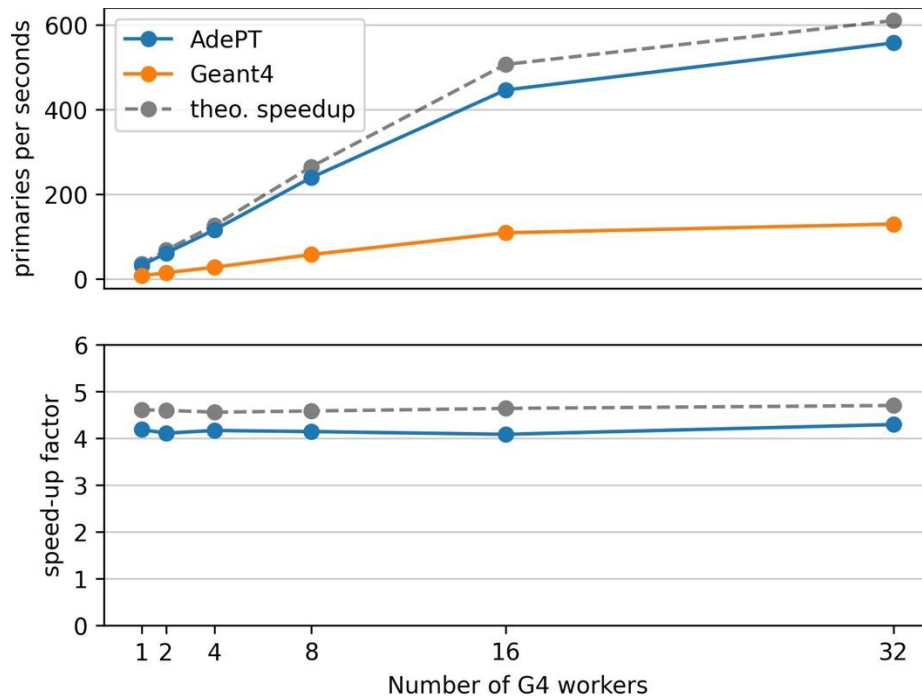


AdePT: current performance results

Application: HGALTB
Geometry: HGAL test beam
No magnetic field
Input: 10k pi- with 30 GeV
Scoring: via sensitive detector code
GPU: **Nvidia RTX4090**
CPU: AMD Ryzen 9 **16 cores**

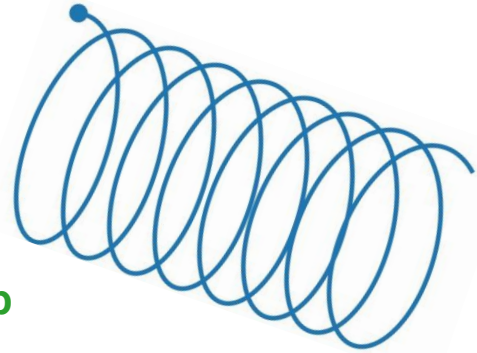


Strong scaling performance



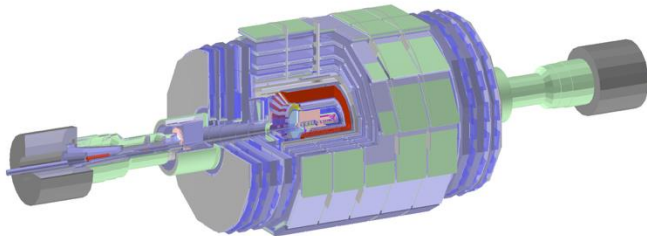
A known bottleneck: propagation in the B field

- Many expensive geometry calls
- Divergence due to different number of iterations
- Long tails of looping particles
 - finishing the tail on CPU gives **~1.4x speedup**
- Reading from field map:
150 MB field map in texture memory via **covfie**
is **only 10% slower** than using a constant number!

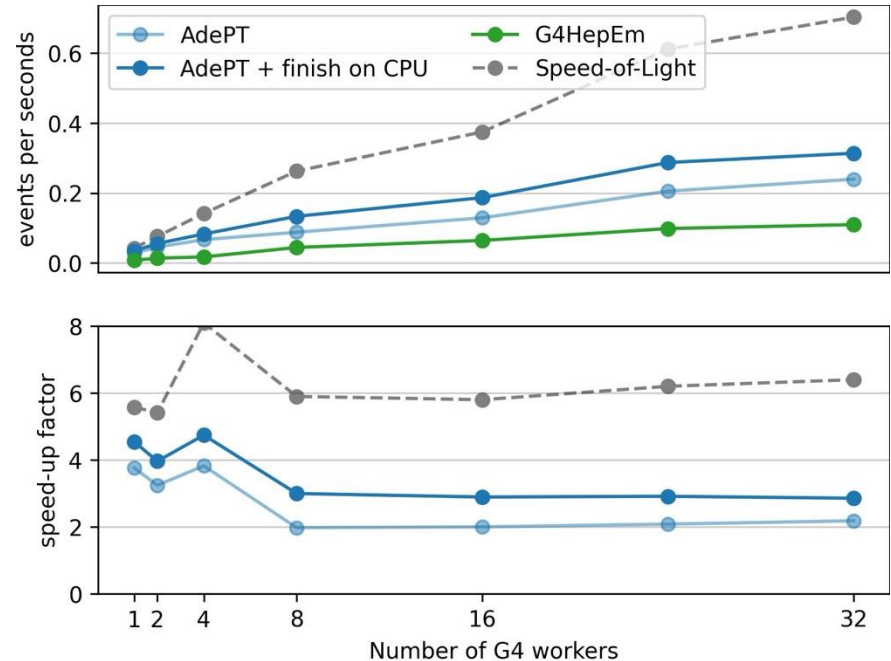


AdePT: current performance results

Application: AdePT example1
Geometry: CMS Run 3
Input: 20 ttbar events per thread
Constant 3.8 T field (from map)
Simple scoring of energy deposition
GPU: **Nvidia RTX4090**
CPU: AMD Ryzen 9 **16 cores**



Weak scaling performance



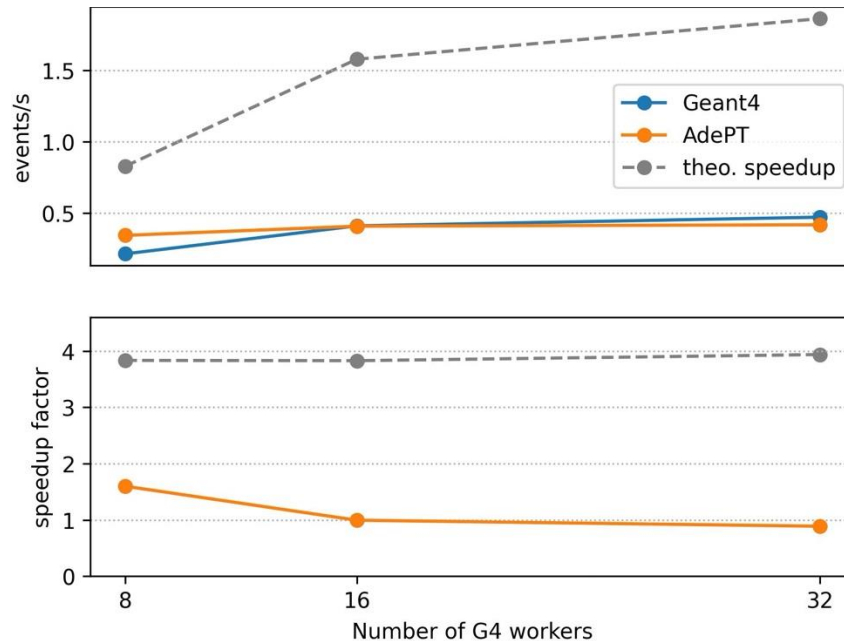
~1.4x speedup in CMS geometry

AdePT: current performance results

Application: FullSimLight
Geometry: full ATLAS calorimeter
Input: 1000 ttbar events
B field: 3D field map
Simple scoring in SteppingAction
GPU: **Nvidia RTX4090**
CPU: AMD Ryzen 9 **16 cores**

Speedup full machine: **0.9x**

Strong scaling performance



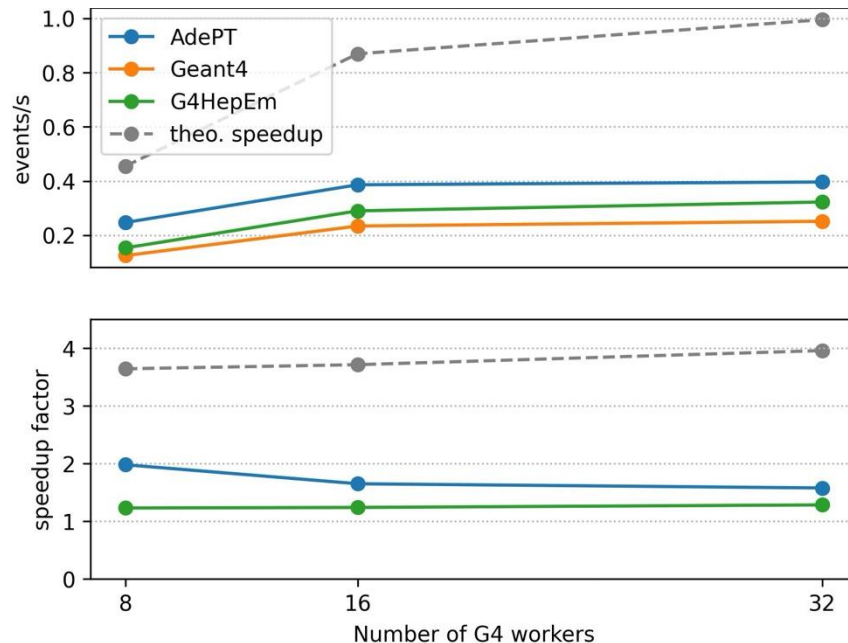
AdePT: current performance results

Application: FullSimLight
Geometry: full ATLAS calorimeter
Input: 1000 ttbar events
B field: 3D field map
Simple scoring in SteppingAction

Hardware: Perlmutter 1/4 node
GPU: **NVIDIA A100**
CPU: AMD EPYC 7763 (16 cores)

Speedup full machine: 1.6x
G4HepEm: 1.3x

Strong scaling performance



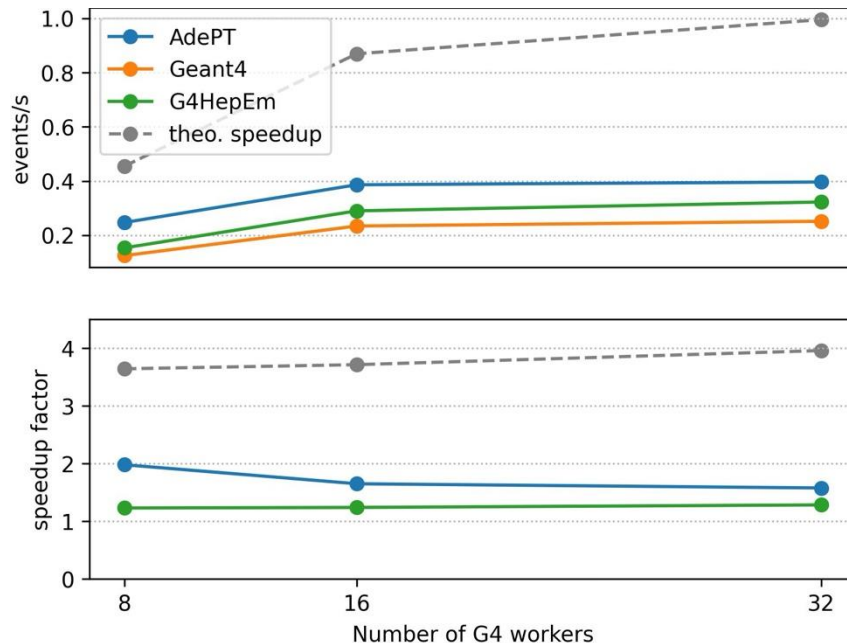
AdePT: current performance results

Application: FullSimLight
Geometry: full ATLAS calorimeter
Input: 1000 ttbar events
B field: 3D field map
Simple scoring in SteppingAction

Hardware: Perlmutter 1/4 node
GPU: **NVIDIA A100**
CPU: AMD EPYC 7763 (16 cores)

Speedup full machine: 1.6x
G4HepEm: 1.3x

Strong scaling performance



Athena has a different run time by factors for production runs. Results cannot be extrapolated!

Current status of LHC experiment simulations with AdePT

Many crucial developments in the last few month, but still missing:

Gauss: **close**

missing G4UserTrackInformation, special cuts on process level

Athena: **medium**

missing G4UserTrackinformation, multi-level B field with currents,
G4DisplacedSolids in VecGeom

CMSSW: **medium**

missing G4UserTrackInformation, custom B field per volume

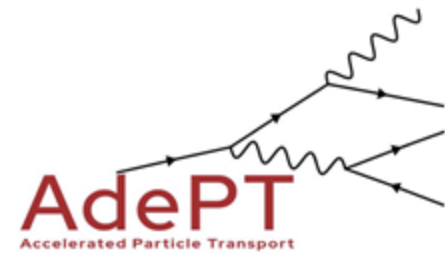
O2: **far**

Missing overlap-priority in geometry, ...

Summary and Outlook

- AdePT offers a **flexible CPU-GPU workflow** as required by the experiment frameworks
- Current performance results are encouraging
but only full production runs in the framework are relevant
- Integration into experiment frameworks has **highest priority** and is ongoing
- Raw performance must be further improved - geometry is the bottleneck:
VecGeom was designed for vectorized CPUs, not GPUs!
R&D for GPU-optimized geometry ongoing

Developers and contributors



Guilherme Amadio, CERN

John Apostolakis, CERN

Gabriele Cosmo, CERN

Andrea Dell'Acqua, CERN

Severin Diederichs, CERN

Andrei Gheata, CERN

Juan Gonzalez Caminero, CERN

Bernhard Gruber, CERN

Stephan Hageboeck, CERN

Jonas Hahnfeld, CERN

Ben Morgan, University of Warwick

Mihaly Novak, CERN

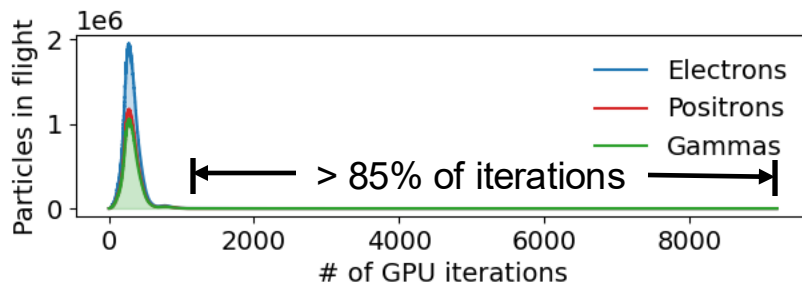
Witold Pokorski, CERN

Juan Rubio, UNAL

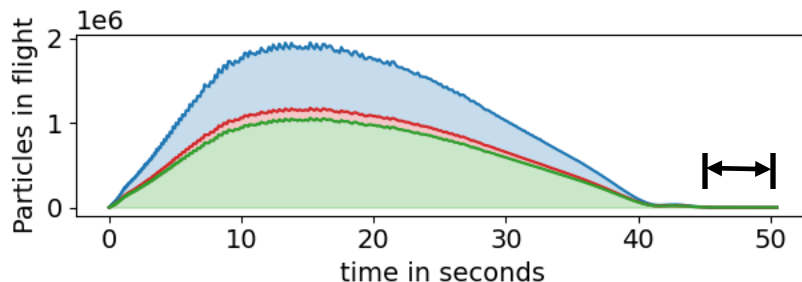
Backup



Tails require many steps but not so much time



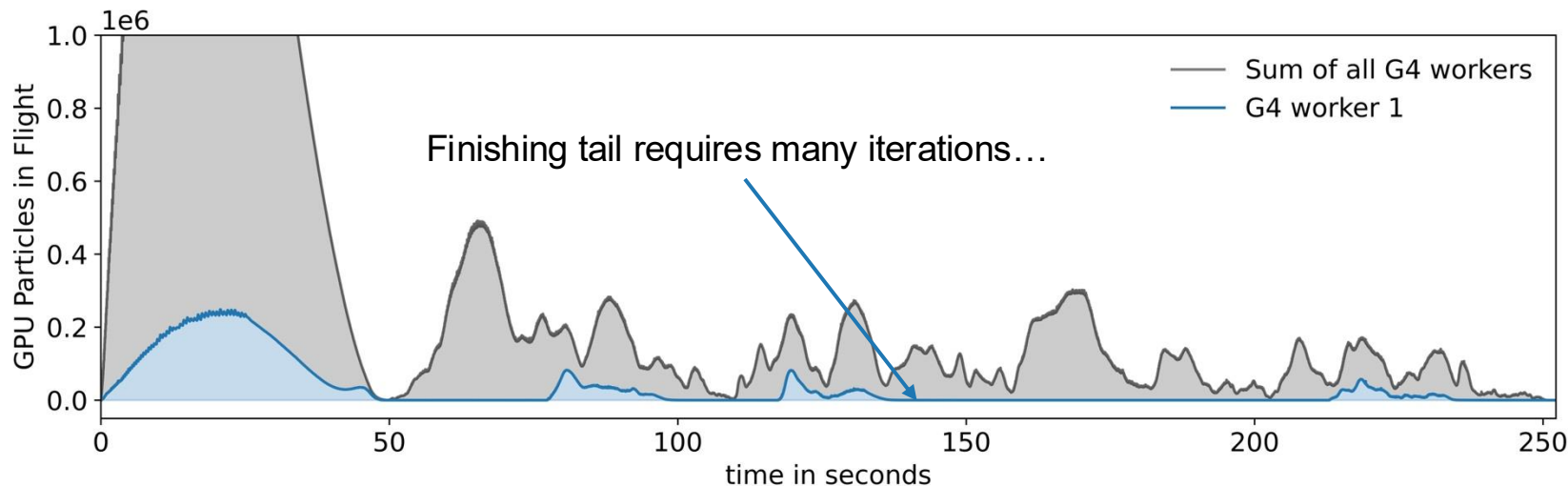
FullSimLight:
16 threads, 16 events



If the GPU is having only few particles in flight...

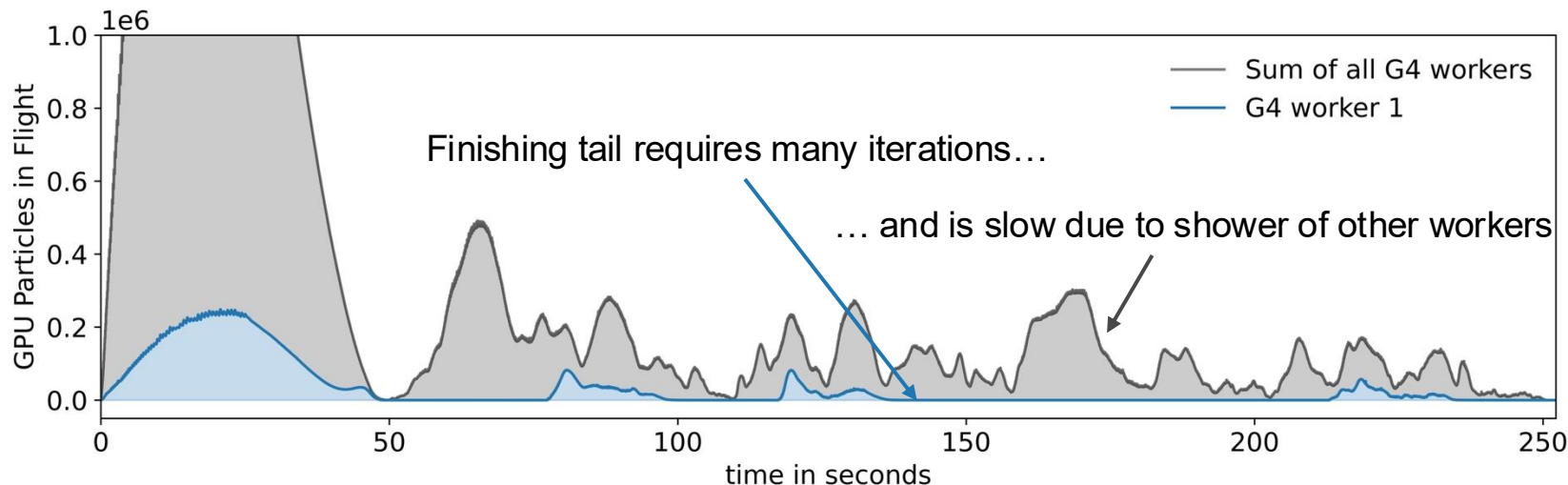
Tails require many steps ~~but not so much time~~ and a lot of time!

FullSimLight:
16 threads, 64 events

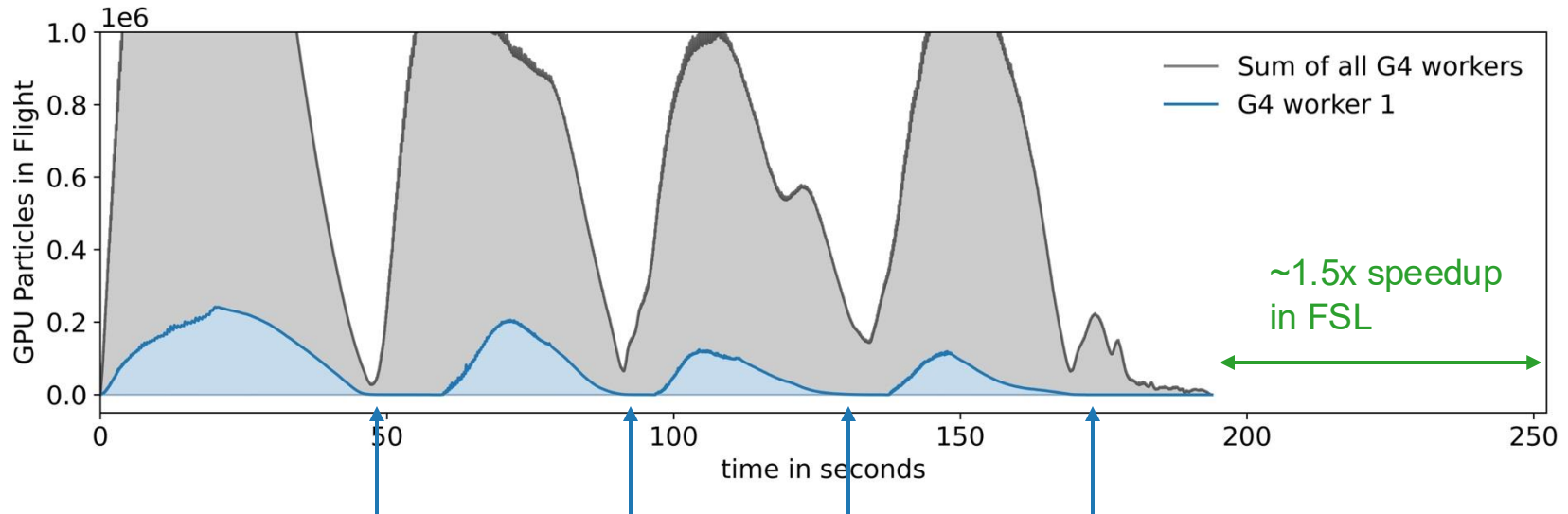


Tails require many steps ~~but not so much time~~ and a lot of time!

FullSimLight:
16 threads, 64 events



Tails require many steps ~~but not so much time~~ ~~and a lot of time!~~ and should be finished on CPU



Push last N (~100-300) tracks from GPU to G4 worker to finish the tail

Difference in UserAction calls

TA: TrackingAction
SS: Step + SteppingAction
S: Step
SA: SteppingAction

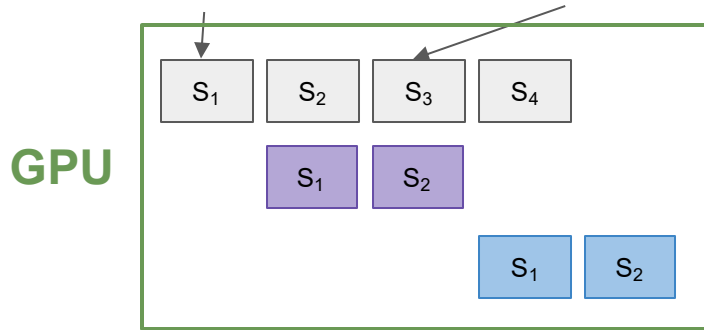
Geant4 tracks **sequentially**:



Difference in UserAction calls

AdePT tracks in parallel:

generates secondary generates secondary

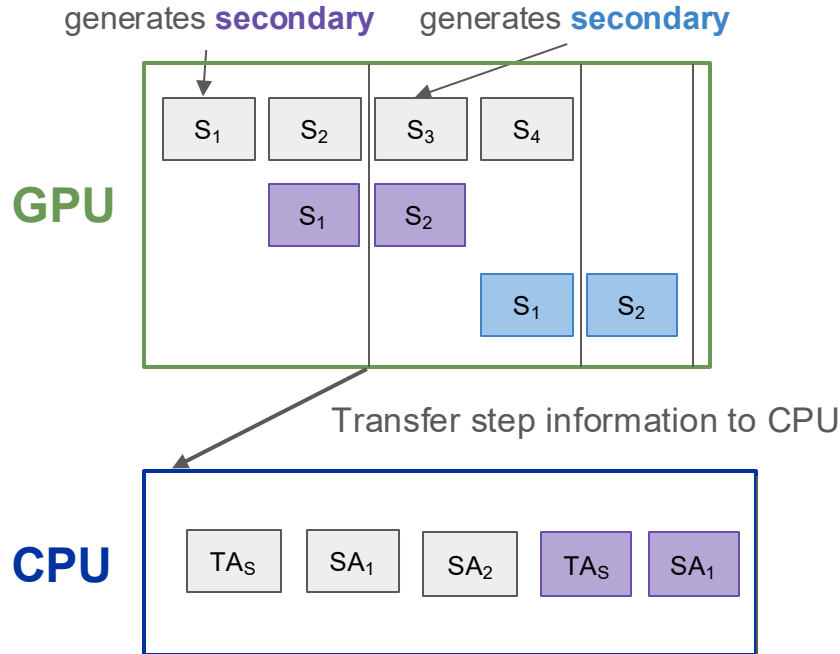


TA: TrackingAction
SS: Step + SteppingAction
S: Step
SA: SteppingAction

Difference in UserAction calls

AdePT tracks in parallel:

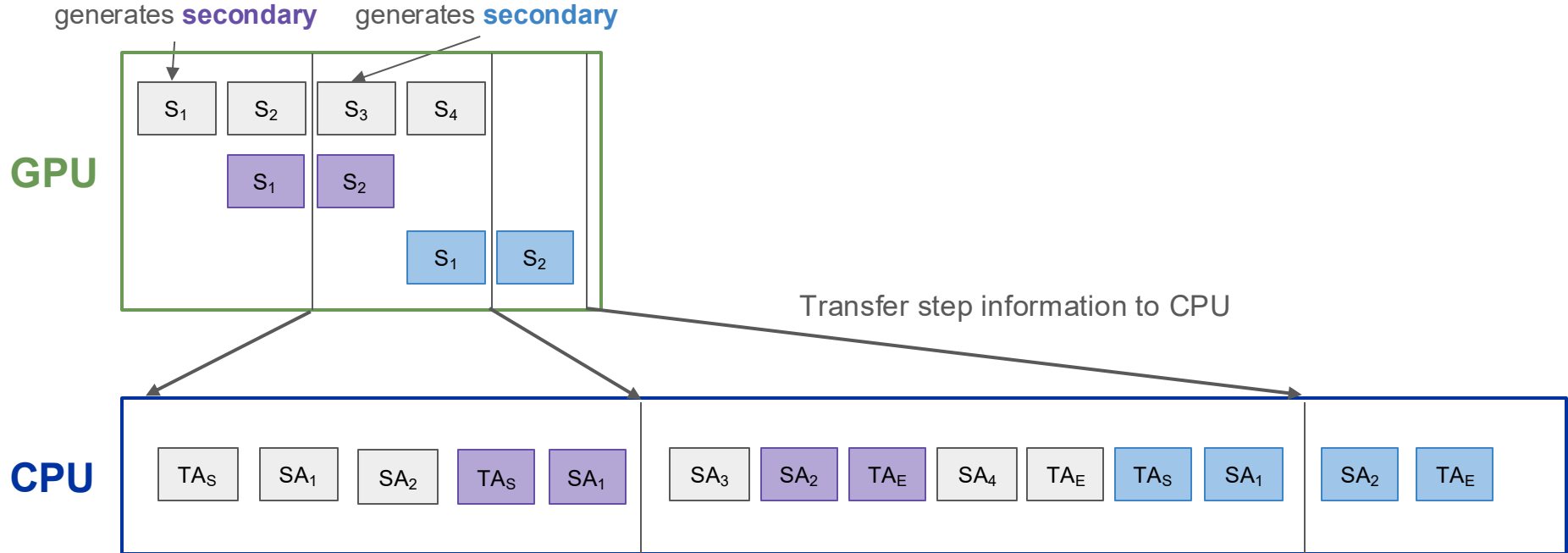
TA: TrackingAction
SS: Step + SteppingAction
S: Step
SA: SteppingAction



Difference in UserAction calls

AdePT tracks in parallel:

TA: TrackingAction
SS: Step + SteppingAction
S: Step
SA: SteppingAction



Difference in UserAction calls

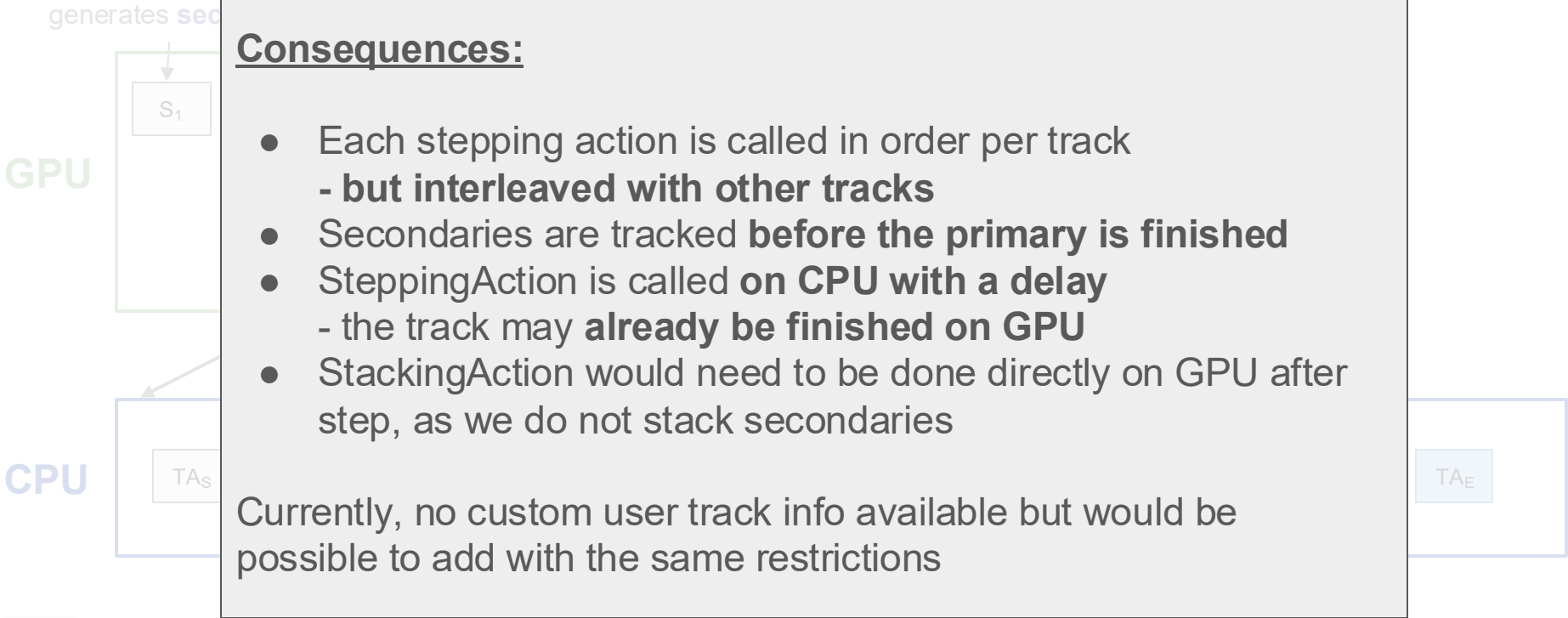
TA: TrackingAction
SS: Step + SteppingAction
S: Step
SA: SteppingAction

AdePT tracks in parallel:

Consequences:

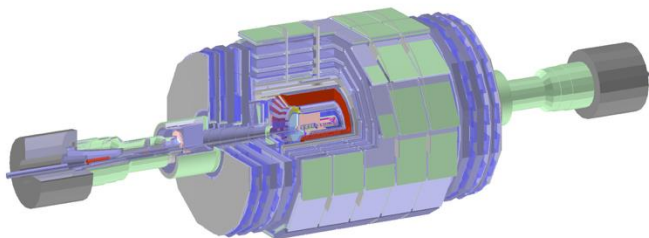
- Each stepping action is called in order per track
- **but interleaved with other tracks**
- Secondaries are tracked **before the primary is finished**
- SteppingAction is called **on CPU with a delay**
- the track may **already be finished on GPU**
- StackingAction would need to be done directly on GPU after step, as we do not stack secondaries

Currently, no custom user track info available but would be possible to add with the same restrictions

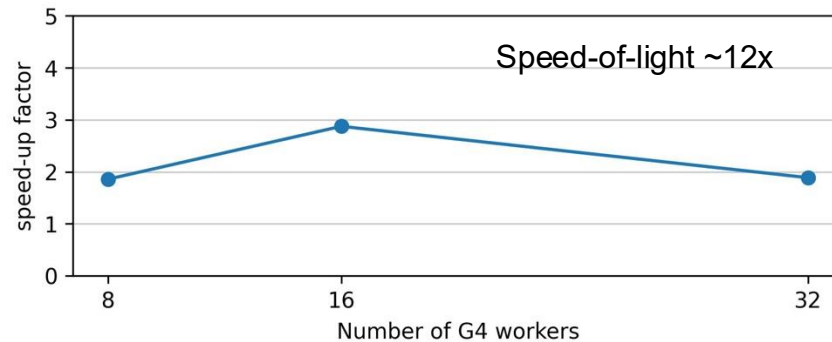
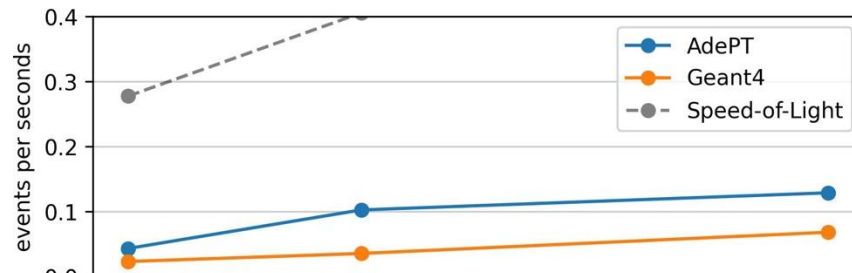


AdePT: current performance results - revisited

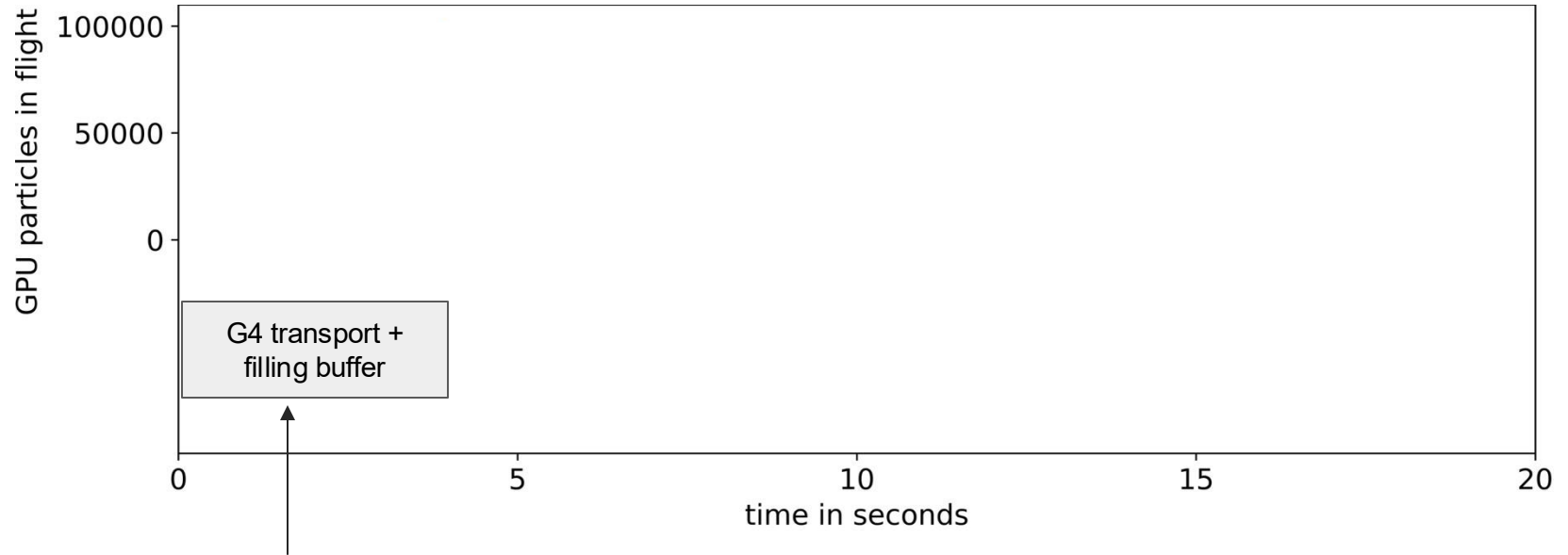
Geometry: CMS Run **4**
Input: 20 ttbar events per thread
Constant 3.8 T field (from map)
Simple scoring of energy deposition
GPU: **Nvidia RTX4090**
CPU: AMD Ryzen 9 **16 cores**



Weak scaling performance

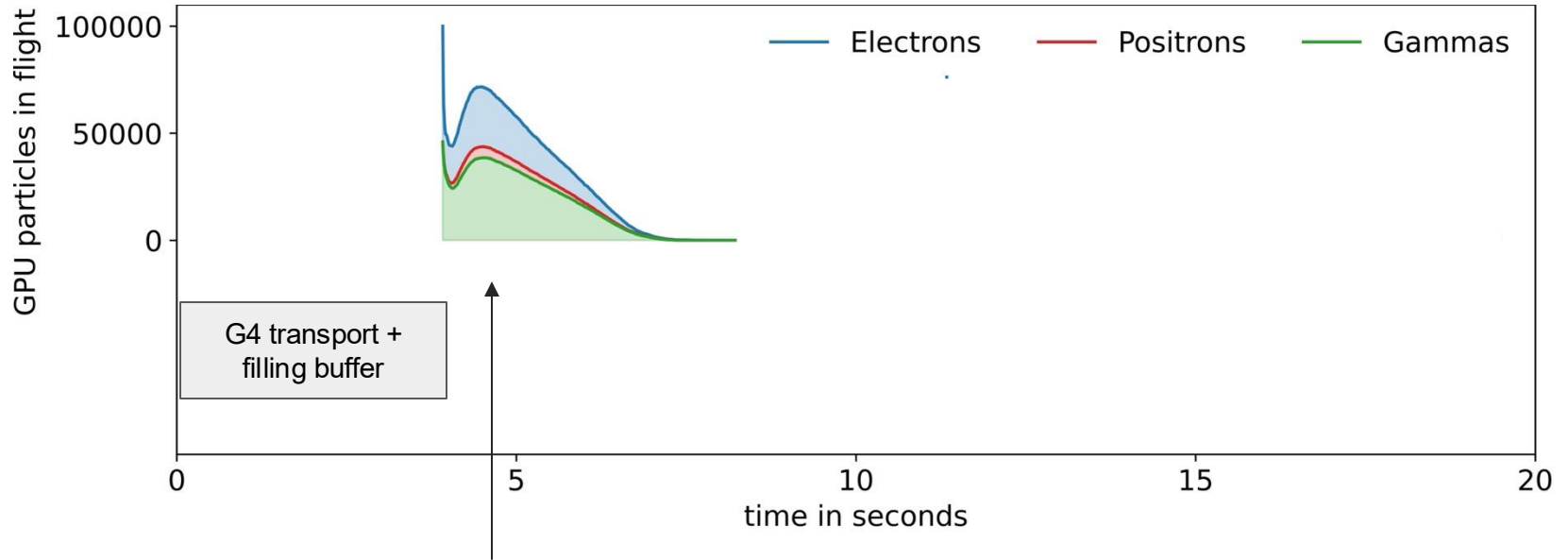


Previous approach: sequential kernel scheduling



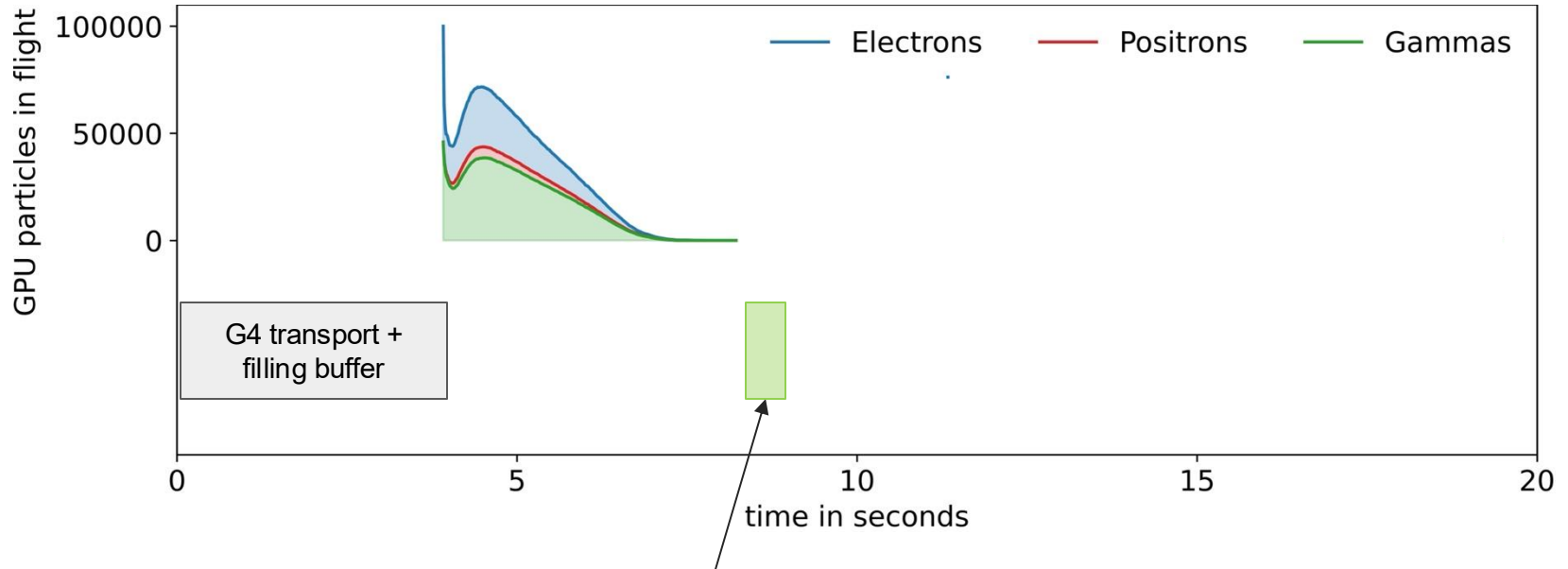
G4 worker: transports particles on CPU, stores e^- , e^+ , γ in buffer

Previous approach: sequential kernel scheduling



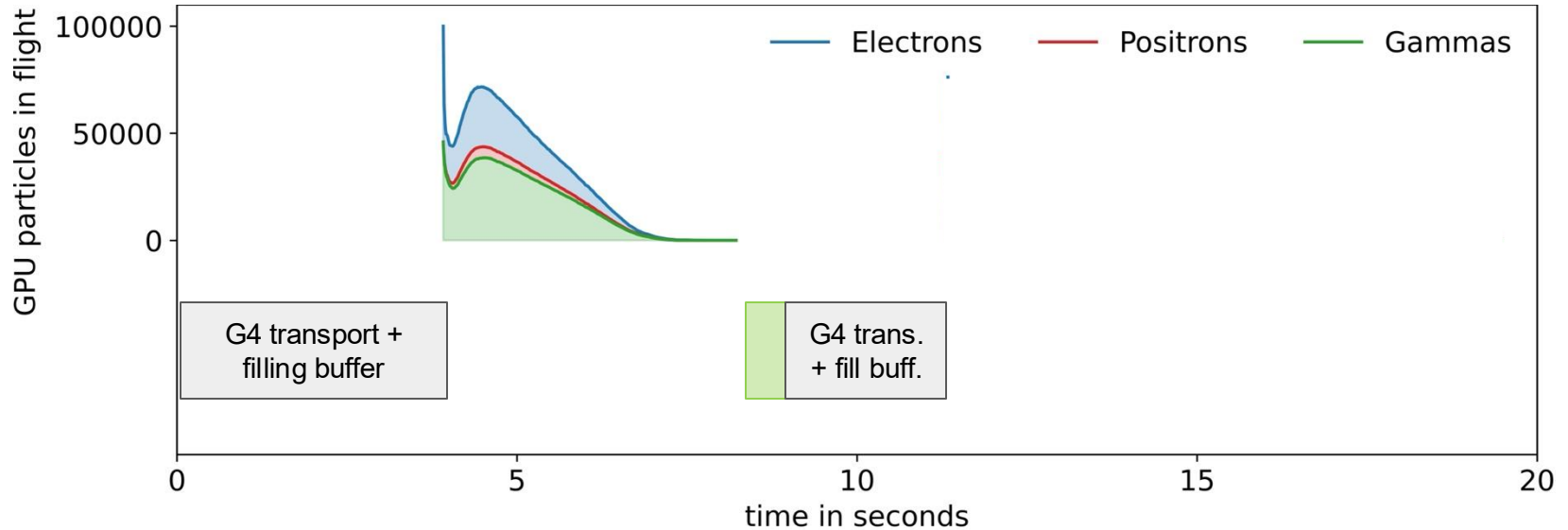
G4 worker: steers GPU transport

Previous approach: sequential kernel scheduling



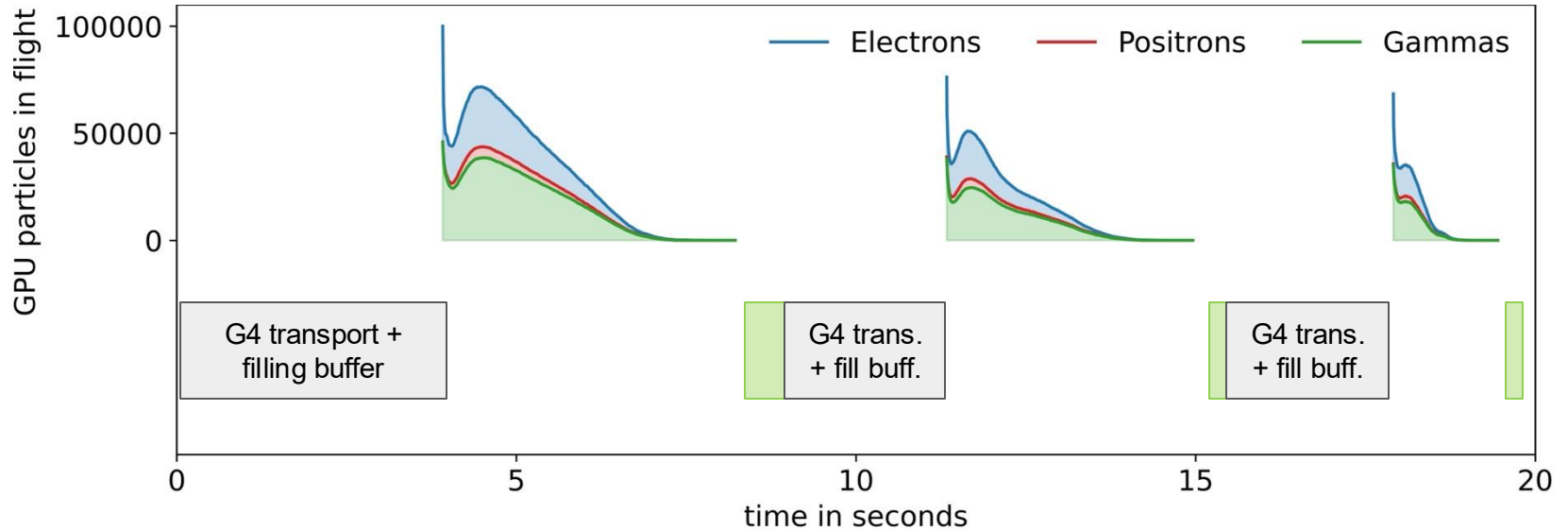
G4 worker: Copy back steps from GPU, call SD code, Tracking- and SteppingAction

Previous approach: sequential kernel scheduling



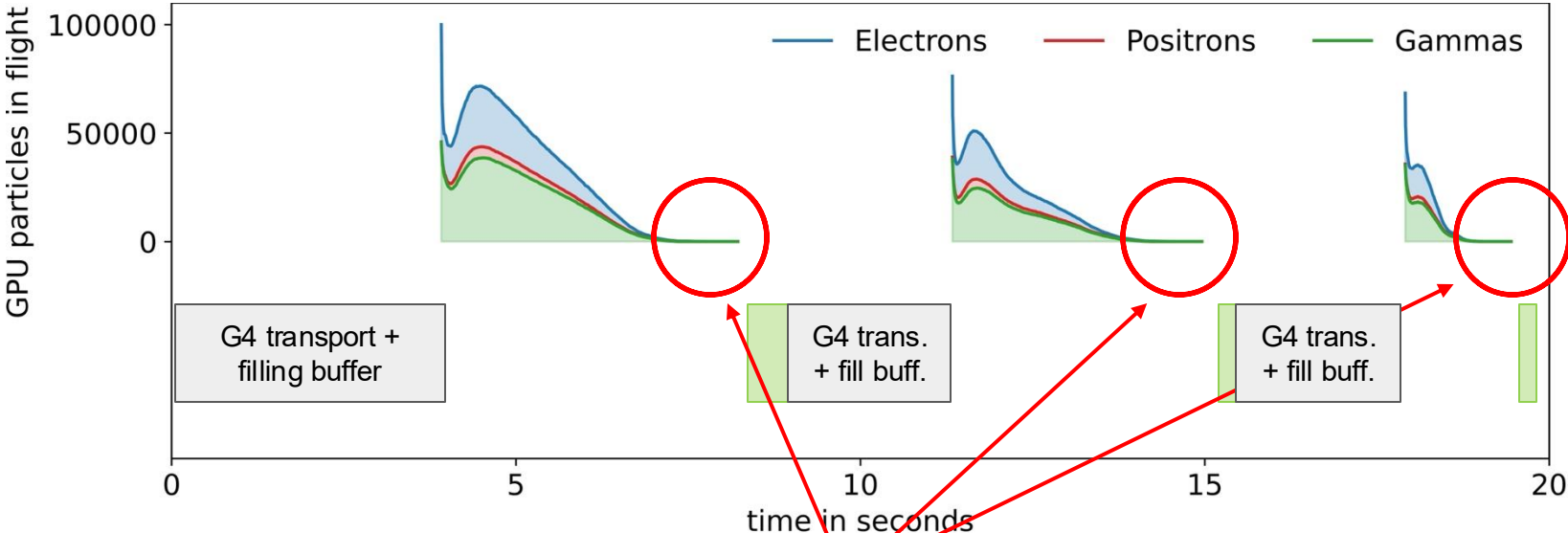
G4 worker: transports particles on CPU, stores e^- , e^+ , γ in buffer

Previous approach: sequential kernel scheduling



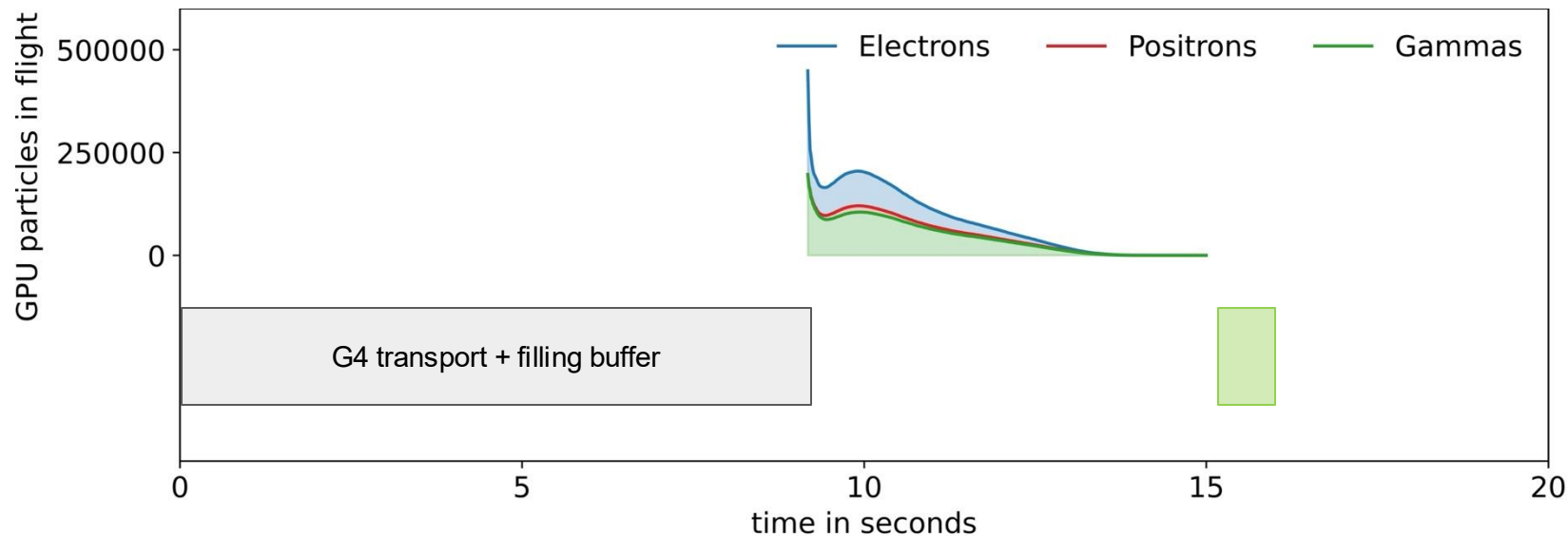
G4 worker: transports particles on CPU and steers GPU *sequentially*

Previous approach: sequential kernel scheduling



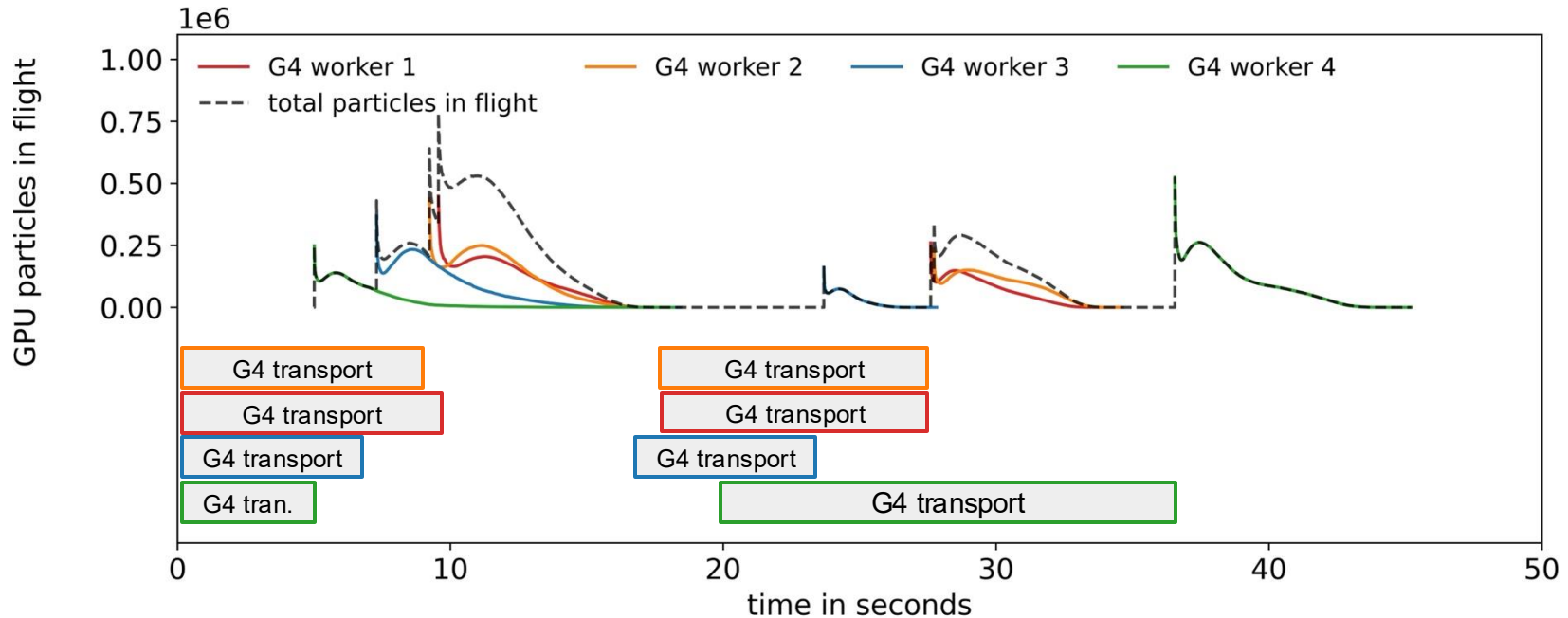
Multiple tails are not ideal...

Previous approach: sequential kernel scheduling



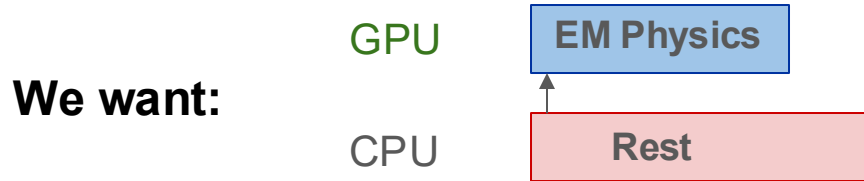
The larger the buffer, the better!

Multiple threads can steer the GPU simultaneously



Each G4 worker fills its own buffer and submits to its own cuda stream
- however, the CPU must still wait for the GPU

The sequential approach has multiple drawbacks



Therefore, the parallel kernel scheduling approach is significantly faster for realistic simulations