R&D directions for performance improvement in Geant4

HEP simulation evolution

Andrei Gheata, Witold Pokorski
EP-SFT

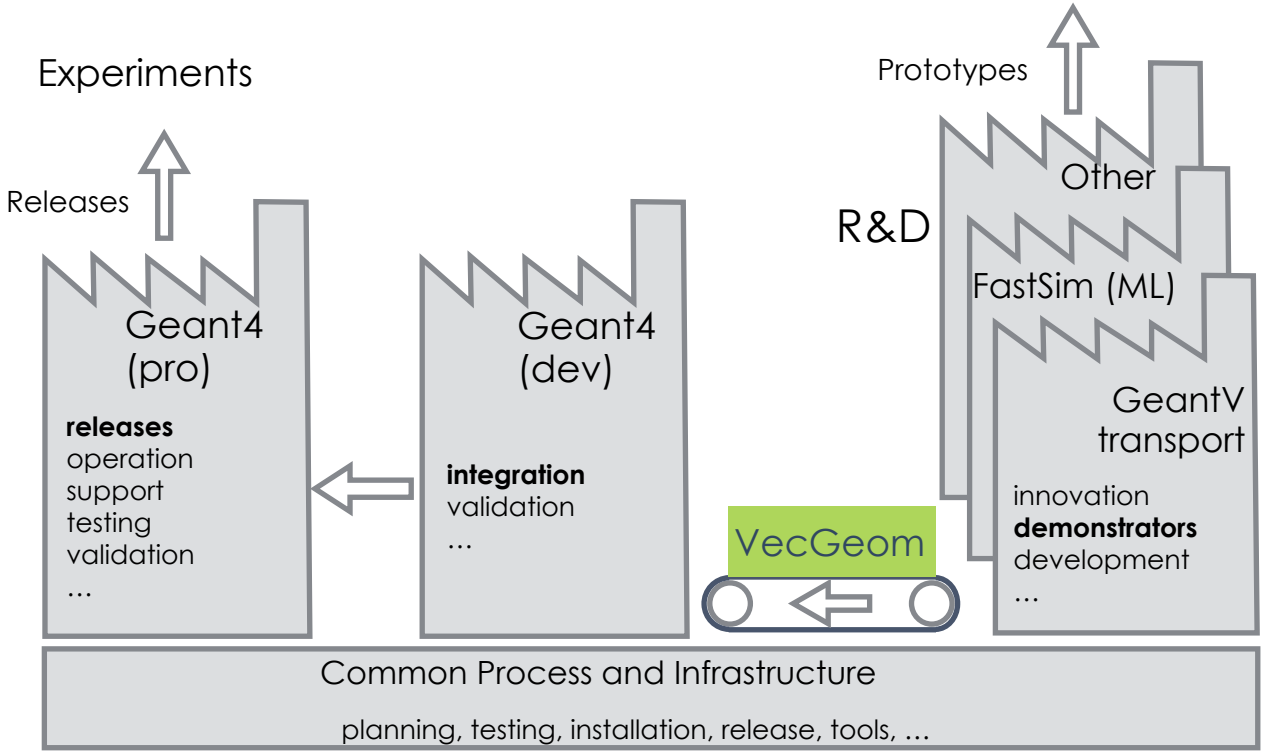
06-07-2020

Main Objectives for Geant4 project

- Provide **production-quality simulation toolkit and support** to experiments
- Provide good long-term **maintenance** while making the toolkit more sustainable
- Improve the **physics models** with better precision and energy range extensions, in particular the hadronic ones
- Improve the overall **computational performance** of simulation

All 4 objectives are of crucial importance and go together

Adopted Strategy

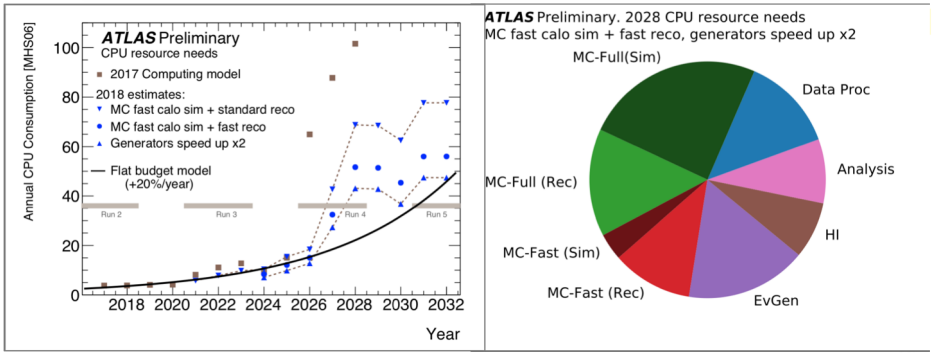


Forecast Simulation Needs

Many physics and performance studies require large datasets of simulated events

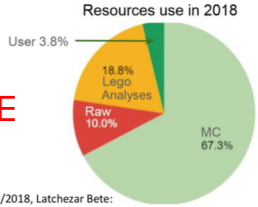
- Geant4 is highly CPU-intensive
- Already lacking statistics -- increasing luminosity poses greater challenges

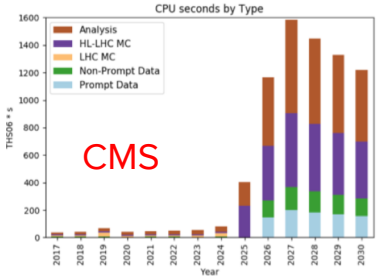
ATLAS



- 2/3 of the computing resources are dedicated to MC simulation, all full sim
  - fast sim not used in production yet
  - fully parametrised fast simulation approach for upgrade studies
- expected 10-100 times more data in Runs 3 and 4
  - cannot cover that with current usage of full sim

ALICE

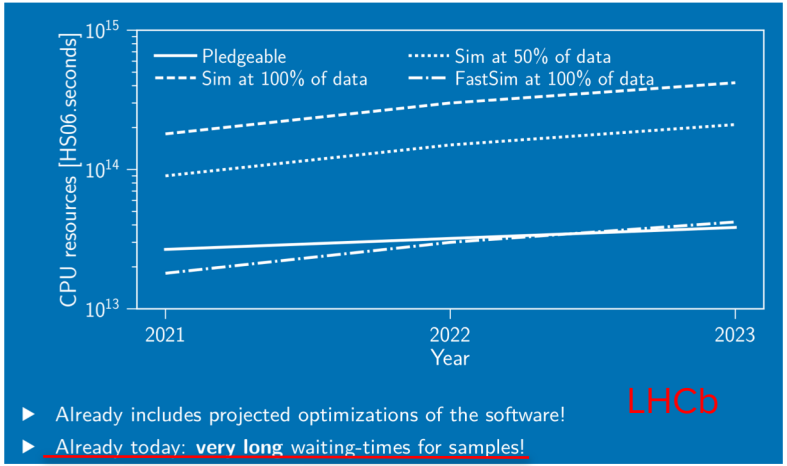




CMS

- Simulate more events to keep up with HL-LHC data volumes: 10×(Phase1)
- May also need to improve accuracy of physics lists to simulate HGCal
- Reconstruction will take longer due to high pileup and granular detectors

➢ Need more events, more accuracy, in more complicated geometry… w/ relatively smaller fraction of total CPU usage



LHCb

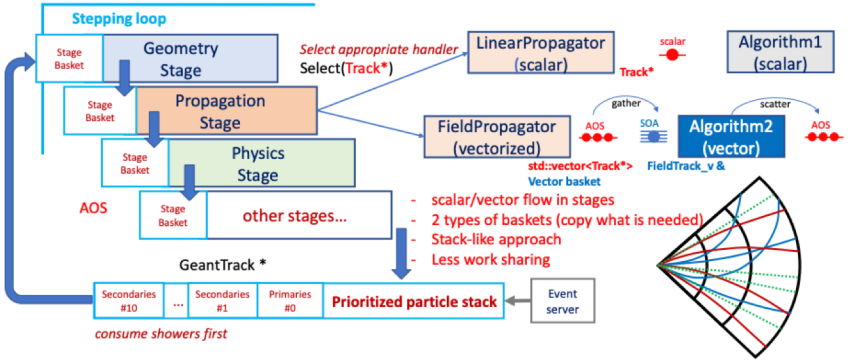
- Already includes projected optimizations of the software!
- Already today: **very long** waiting-times for samples!

Geant4

- Geant4 code written in late 90’s with flexible physics modelling exploiting layers of virtuality
  - allowed progress in physics by comparison of competing models in the same energy ranges and complementary models to be combined to cover large energy ranges.
  - delivered a physics precision that allowed the experiments to perform analysis of the collected data and to produce new results pushing the boundaries of our understanding in HEP
  - most of the LHC physics program relies on this toolkit
- physics models need to continue to evolve to prevent systematic uncertainties from simulation to become a dominant factor
- code of the toolkit is becoming old, with some parts written almost thirty years ago, making it more and more difficult to run efficiently on modern computing architecture
  - overall CPU performance of the current code will not improve drastically just by relying on modern hardware or better compilers

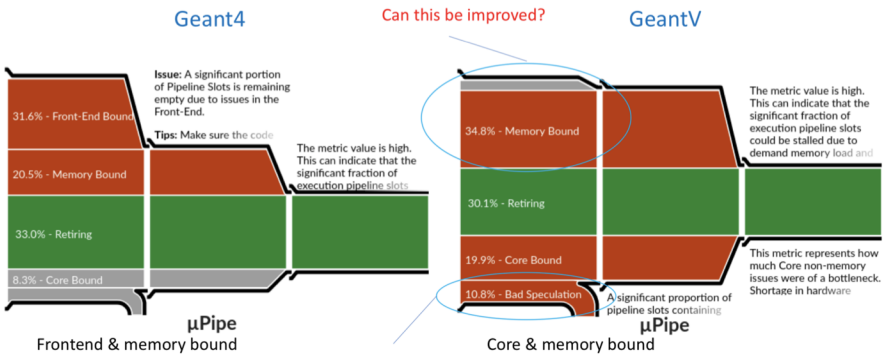
GeantV R&D

- GeantV R&D explored vectorized particle transport for next generation simulation toolkit
  - aimed at demonstrating the speed up of simulation using a novel approach to concurrent processing and data



- HSF community meeting (October 2019) with the outcome of the GeantV prototype
  - Prototype with EM physics was finalised. Many comparisons with equivalent Geant4 application done. Technical paper published in arXiv with all the details of the study.
- Libraries developed for the prototype are very useful (e.g. VecCore, VecGeom, VecMath)
  - Successfully integrated in Geant4, ROOT, etc.

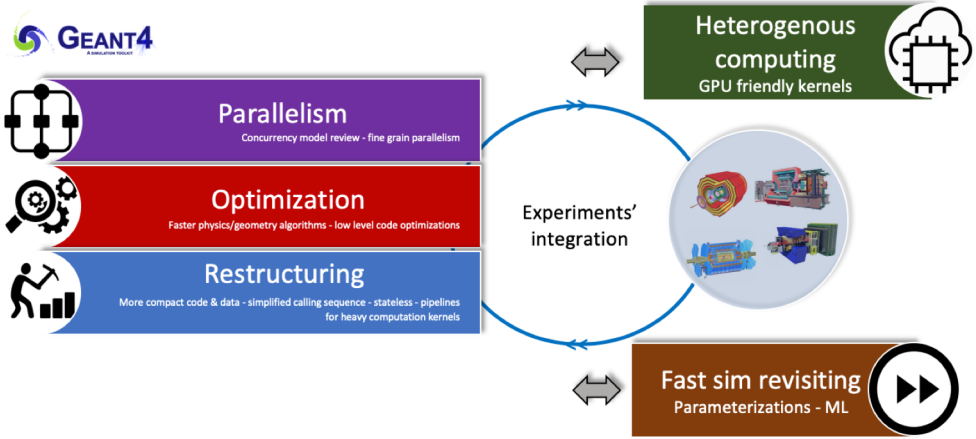
Main lessons from GeantV



- Main factors in the speedup seem to include
  - Better cache use (GeantV single track mode shows performance decrease for small caches)
  - Tighter code (e.g., less classes, indirections and branching), better use of instruction cache
- Vectorization's impact (much) smaller than hoped for and does not bring the expected speedup
  - Small fraction of the code could be vectorized or is run in vector mode effectively
  - Overhead of basketization cost similar to vector gain for "small" modules
  - Basketization can bring benefits for FP hotspots (e.g. magnetic field, multiple scattering)
- Basketization cost in
  - Either extra memory copy (using collection of tracks)
  - Or lower memory access coherency (using collection of pointers)
- Re-writing and modernising large parts of Geant4 potentially could bring several tens of % speed-up (depending on the CPU/caches)
  - Compact code, better data formats, data locality, less virtual functions, etc.

Three Main Axes of Development

- **Improve, optimise and modernise** the existing Geant4 code to gain in performance for the detailed simulation
  - Re-structure the code to make possible major changes (task-oriented concurrency, specialisation of the physics, better data formats, etc.)



- Trade precision for performance using **fast simulation techniques** both with parameterisations and with ML methods, and integrate them seamlessly in Geant4
  - Use detailed simulation to 'train networks' or to 'fit parameters' that later can deliver approximative detector responses well integrated within Geant4
- Investigate the **use of accelerators** such as GPUs
  - With novel approaches for organising the computational work

Geant4 R&D Task Force

- **Created to promote and survey research activities** on potential software architectural revisions and the exploitation of emerging technologies or computing architectures that would be beneficial to Geant4
- **Ensure the visibility** of these R&D activities inside and outside of collaboration
- Where appropriate, conduct benchmarking comparison and **provide/assist communication and support among the R&D activities**
- Make timely **assessment reports to the G4 Steering Board** with solid proof of benefits and eventually spawn dedicated task forces to create workflow, estimate required resources and drive that particular development for integration into the code base

Improve, optimise and modernise Geant4

- Ongoing development in all Geant4 Working Groups
  - from simple technical improvements, through the code modernization (more compact) to new architectural choices
- Few selected topics
  - Sub-event parallelism
  - Task based Geant4
  - VecGeom navigation
  - Transportation and physics ‘framework’ review

Sub-event parallelism and tasking framework

- New “tasking” framework for Geant4
  - Pool of threads without a predefined call-stack
  - Tasks are essentially function calls that are placed in a queue
  - Threads in pool are idle until tasks are placed in the queue

Available as an option in release 10.7-beta

- Naturally allows fine-grained parallelism
  - Every G4Track could a be ‘task’
  - Could play a critical role in how Geant4 could leverage the GPU
    - one or more thread-pool instances per master-thread, “task groups", and CUDA streams
      - balanced workload between the CPU and the GPU
      - nearly-perfect scaling on the CPU is maintained in addition to computation boost from the GPU

Jonathan R. Madsen
✉ jrmadsen@lbl.gov
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory

VecGeom navigation in Geant4

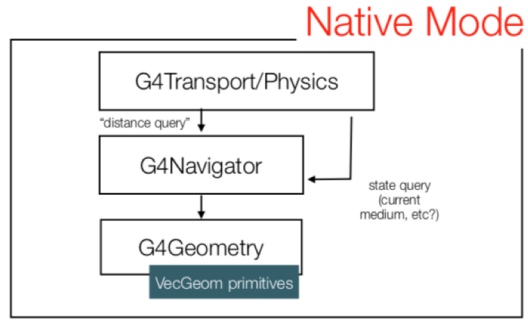
- VecGeom is evolution of G4/TGeo/Usolids geometry with the goal to
  - use SIMD acceleration as much as possible (multi-particle API, single-particle API)
  - modernize / revise / optimize algorithms in general

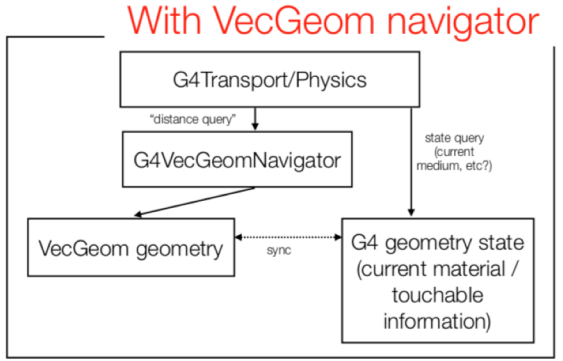
- Two main components provided for use in simulation:

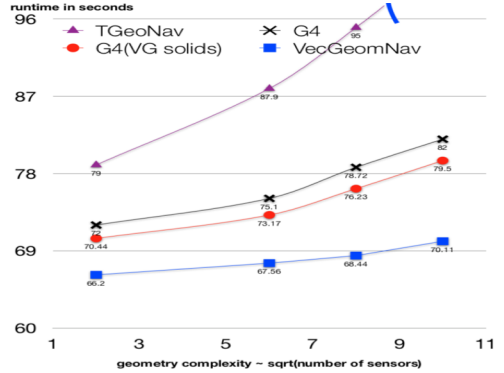
  - elementary and composite **geometry primitives** (box, tube, ...)
  - geometry modelling + **navigation**
    - First performance benchmark transporting 100K primary electrons as a function of geometry complexity (number of sensors) yields encouraging numbers

- The VecGeom navigator plugin **accelerates overall simulation time**
  - Here between 8 and 17% compared to plain G4

Available as an option this year.







J. Apostolakis, G. Cosmo, S. Wenzel

Alternative e-/e+ and gamma transport

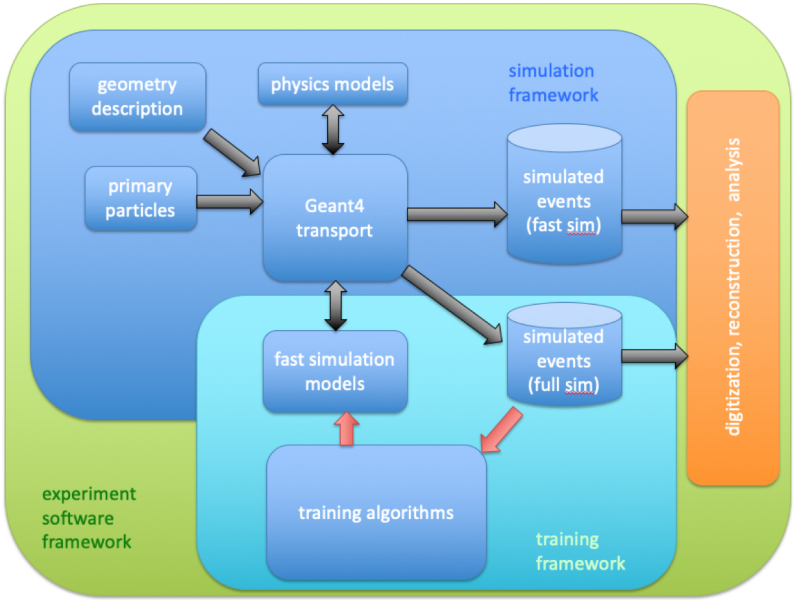
M. Novak

- current simulation framework used in Geant4 (process interface, track, stepping algorithm, etc.) is completely general
  - provides a high level of flexibility
  - brings many potential sources of inefficiency
    - well defined simulation problems such as HEP detector simulations require a small fraction of these functionalities
- new R&D on simplified physics simulation "framework", tailored specifically for HEP detector simulations
  - targeting performance critical particles and their interactions with the highest level of specialization for HEP usage
  - highly reduced complexity and size of the corresponding code, more cache efficient data storage designed by considering the run time access pattern, etc
  - by creating a separate, self-contained part of the simulation provides a natural starting point for GPU-related R&D where the corresponding work is performed on the device

Fast Simulation Revisiting

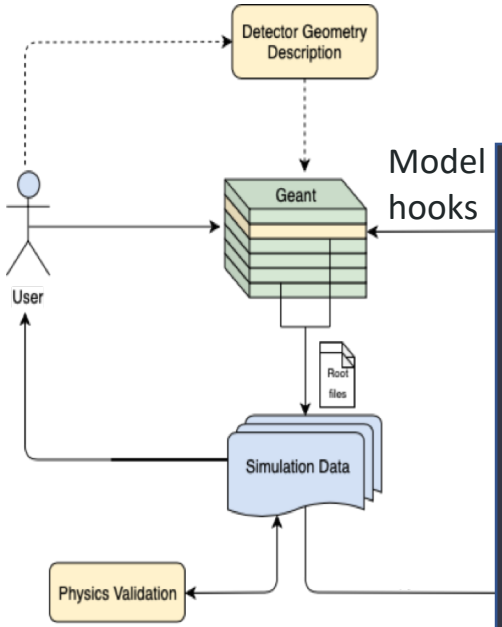
- ‘Fast simulation’ techniques (non stepping-based) are essential for achieving the required statistics
  - seamless integration of ‘full’ and ‘fast’ simulation is a requirement on the simulation toolkits

- several approaches to fast simulation studied
  - improved (automated) parameterizations
  - Machine-Learning based tools
  - biasing techniques



Automated tools for fast simulation

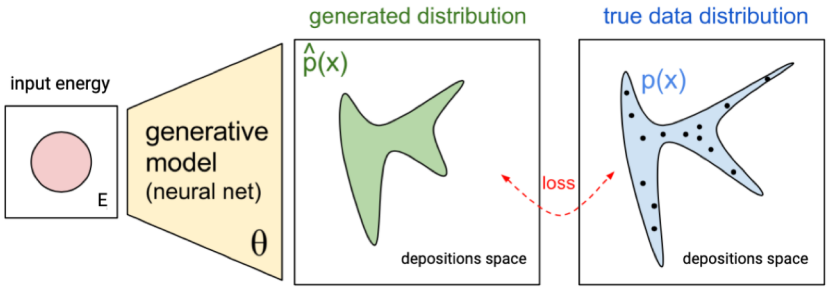
- Fast simulation is experiment dependent
  - custom procedures to extract parameterisations
- Ongoing work for streamlining the procedure
  - Users come with their own setup
  - Full simulation producing standard information (hits)
  - Simplified, automatic and easy to use procedures to extract the parameters (e.g. fitting shower profiles, training networks)
  - Plugged back in simulation via Geant4 fast sim hooks
- Goal: improve precision of fast simulation models

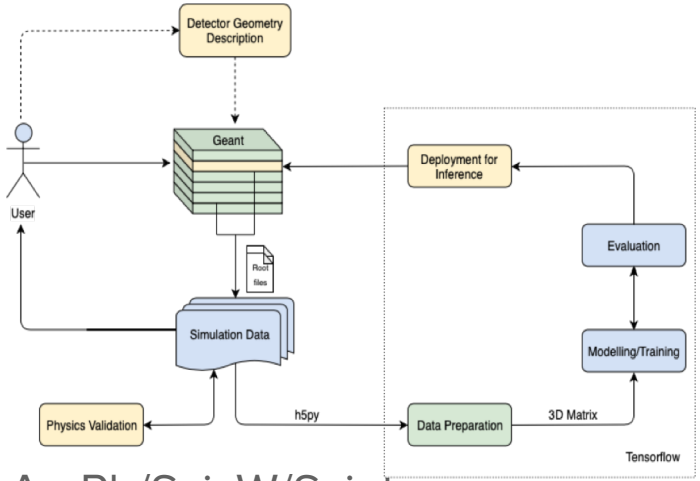


- Tools for extracting parameters of fast simulation models
- Tools for training and inference for generative models

I.Ifrim, W. Pokorski, A. Zaborowska

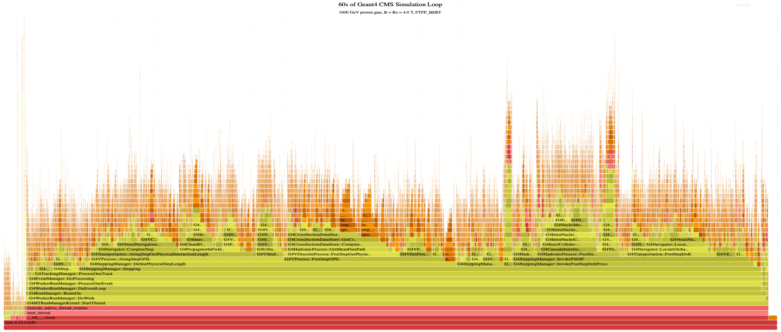
Machine Learning Approaches





- Training on simulated data
  - Different calorimeter types : PbWO4, Pb/LAr, Pb/Sci, W/Scint
- Different architectures are tested
  - Autoregressive, VAE, GAN
  - Validation against MC truth
- Training/inference workflow aiming to integrate with Geant4 hooks

Investigate Use of Accelerators



- General HEP simulation code is **not a natural candidate to run on GPUs**
  - Large complex codes, computation spread in many areas, many branches and special cases
  - Work needed to be done not known a priori (stochasticity)
  - Amdahl's law not helping here (any sequential code in CPU will limit maximum speedup)
- Some successes on reduced and simpler problems
  - Low energy electromagnetic, medical app. with simple geometries, neutrons transport, etc.
- There may be other alternatives (non-GPU based HPC), but GPUs are certainly widely available
  - Pressure from funding agencies to make efficient use of large HPC installations
  - We won’t get the necessary speedup by running on CPUs
- Big issue on the sustainability of the code
  - No standard GPU programing language

GPU R&Ds

- Goal: transform the very heterogenous Geant4 HEP particle transportation into a more homogenous computational problem

- Development of representative ‘demonstrators’
  - Before embarking in a massive conversion and re-engineering of large parts of the code we need to demonstrate its feasibility

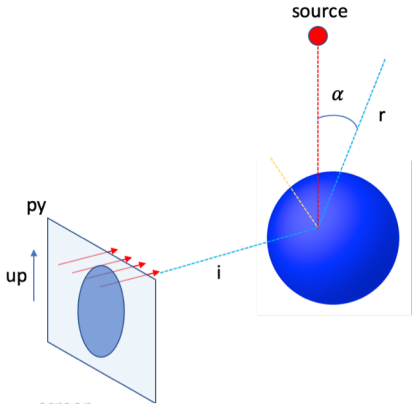
- Study sustainability aspects of the code and the use of portability libraries
  - Gain experience in using libraries such as Kokkos, Alpaka, etc.

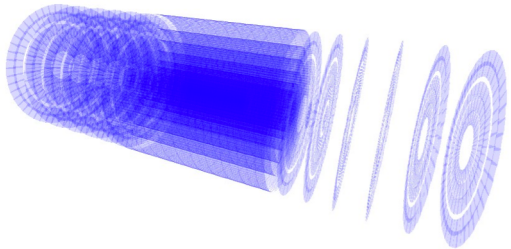
Prototyping simulation workflows on GPUs

- Geant4 not portable as is on GPUs
  - need to develop simplified prototype, adding more and more components to understand how to tackle the problem
- Geometry: an important component in such simulations
  - accelerator-aware geometry library: VecGeom.
    - Code is cuda-annotated using macros, then compiled in separate libraries for both host (gcc, clang, ...) and device (nvcc).
- Building raytracing code running on arbitrarily complex geometry to start with, incorporated for now in VecGeom itself

Ray tracing prototype current goals

- Can we navigate VecGeom-based geometry on GPU? (Yes)
  - Complex geometry, same code on CPU/GPU, single kernel for pixel scoring, rays represented as POD and handled like tracks
- Are there limitations? (Yes)
  - Geometry complexity, data size, stack size
- How can we handle generating rays from kernels?
  - Add extra complexity layer handling reflection/refraction
- What is the GPU usage for such a simple model?
  - Is it possible to improve? How?





Next steps: exploring more complex setups

- Adding new components
  - Field propagation, simple physics models
- Scheduling a pipeline simulation using these components
- Scoring detector data
- Combining CPU/GPU flow
- Understanding to which level a framework/toolkit approach is still feasible

γ and e+/- transport using GPU vendor libraries

- proof of concept simulation using a GPU-vendor library such as Nvidia Optix for the full transport
  - alternative or complementary to VecGeom approach (discussed on previous slides)
- Examine the existing Opticks simulation of optical photons, to identify similarities and differences with operations needed for interactions of gamma and/or electrons;
- Map interactions into shader operations of NVidia Optix, in order to do the full transport of gammas on GPU;
- Create a prototype implementation of transport of at least one particle species (gammas and/or electrons) leveraging a vendor library, likely Optix.

Conclusion

- Without significant improvement in precision and speed, simulation will become a showstopper as far as new physics studies are concerned
- Several activities ongoing to modernize Geant4 code, however a major R&D needed to achieve required speed up
- Recent studies indicate that in order to try to use the potential of accelerators, deep architectural changes (in tracking flow, geometry) need to be investigated
- What is already clear now is that **major effort will be needed** to ‘evolve’ the Geant4 toolkit to cope with challenges of HL-LHC
  - input and collaboration with experiments is essential here
    - fast simulation techniques, GPU usage