Applying vectorisation to Geant4 (and few other R&D items)

Andrei Gheata

23rd Geant4 Collaboration Meeting

Lund, 27-31 August 2018

Motivation

- Some of the R&D subjects followed by GeantV can be applied/bring benefits in Geant4
  - Single particle mode: better algorithms, vectorization on internal loops (transparent to users)
  - Multi-particle mode: some important hotspots can profit from basketizing **w/o** having GeantV-like scheduling (small impact for users)
- Some R&D subjects can bring benefits to both scalar/vector approaches: optimizations
  - Study usage of single precision in some algorithms: major benefits for SIMD algorithms, half memory bandwidth for all cases
  - Topology-aware navigation (geometry), usage of multiple physics lists, caching expensive computation results (e.g. logE), other low-level optimizations
- Extending the multi-particle transport approach to other parts of Geant4 would allow for more important gains
  - But would require deeper structural changes at scheduling level introducing also important perturbations for the users

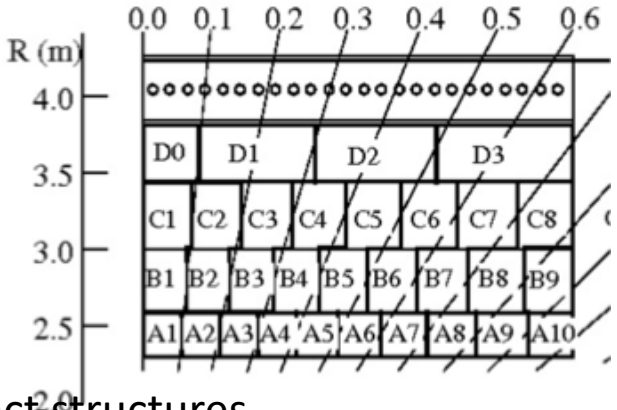
Geometry optimizations

- Geometry navigation handled on multiple levels
  - (Global)Navigator -> volume optimizers -> solid algorithms
- Different tasks involved
  - Distance in current material, isotropic safe distance, relocation after crossing
  - Some of these tasks can be optimized both in context of single and multi-particle transport
- Using VecGeom native navigation in Geant4 (see Sandro's presentation)
  - Improved volume optimizers: bounding volume hierarchy vectorized helpers
- Using specialized volume navigators
  - Pre-computed awareness of neighbor topology and caching of transformation matrices. Specialized code can be generated and in some cases generic.
  - Improve geometry time overall, reduce scalar bottleneck in multi-particle approach

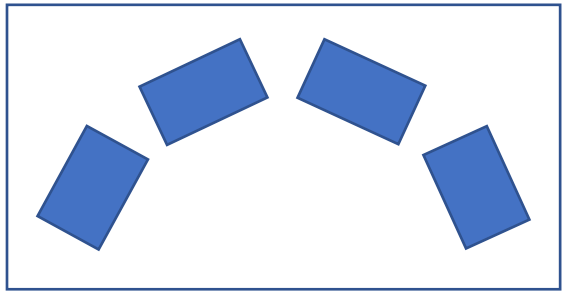
Neighbor topology discovery

- In most cases the number of neighbors is small
  - Sometimes the neighbors are the same (replicated structure)
  - Compact structure: never exiting to mother, but to one of the neighbors
  - Sparse structure: always exiting to mother (1 level up)
- Detection can be done in a “training” phase
  - Using sampling or collision detection algorithms
- Navigators per volume rather than global
  - Already the design in VecGeom
  - Generating a specialized navigator per volume/topology

Repeated structures
(caching matrices)



Compact structures
(exit to neighbor)



Sparse structures (exit to mother/never directly entering daughter)

Specializing navigators

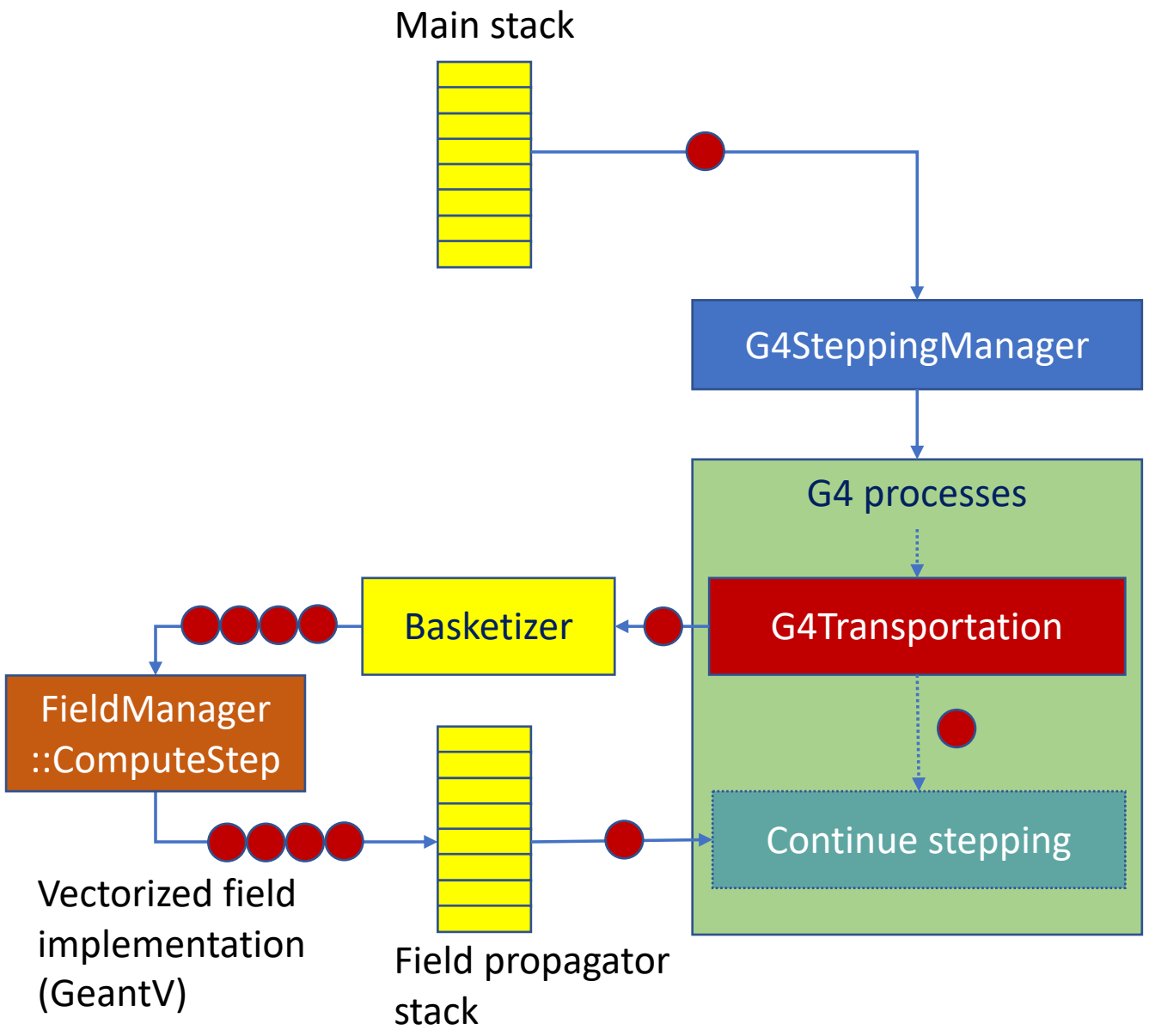
- A first attempt already done by Sandro
  - Experimental for now but available in VecGeom master
- Procedure in 2 steps
  - Sample points/directions in geometry, propagate/relocate after crossing, establish list of neighbors/matrices, generate code including cached matrices, create new library
  - Load navigators from generated library, attach to volumes in a normal session
- Things that can be improved
  - Sparse topologies: no need for code generation
  - Using collision detection to exclude from start certain relocation paths when exiting a certain volume
- Expected performance improvement: order of tens of percent for geometry total time.
  - Prerequisite: VecGeom native navigation in Geant4

Multi-particle treatment in Geant4

- Gathering particle states for concurrent execution of an optimized CPU-intensive algorithm
  - “Basketizing” – central idea of GeantV
  - Efficiency demonstrated already for some parts of the processing so far
    - Propagation in field, MSC, work ongoing for geometry and final states sampling of physics models
- Can it be done (at least partially) in the context of Geant4?
  - **Yes, with sizeable benefits**, even if not profiting from the improved locality given by GeantV-like scheduling
  - Cannot be extended to the full flow without particle-level parallelism scheduled a la GeantV
- First obvious candidate: magnetic field propagation
  - Giving ~10% overall speed-up in GeantV CMS application

Possible approach

- Intercept charged tracks during transportation process
  - Basketize before FieldManager::ComputeStep
- After basketizer, copy track info relevant for field propagation to SOA
  - charge, position, momentum
- Dispatch to vectorized field propagator, then gather outputs to original tracks
- Stack tracks for further sequential processing
- Extension to MSC possible, but more complex, handling multiple basketizers needs GeantV-like scheduling
- User implications on track sequencing to be discussed



Using single precision + vectorization

- Using float very relevant for vectorized code, less for scalar
  - Double number of concurrent operations (vector)
  - Less memory traffic, better cache performance (vector & scalar)
- Input data to algorithms is often known with low precision
  - Geometry parameters, cross sections
  - Simulation would “survive” the precision loss if doing most computation in single precision
- Many algorithms have however some very sensitive parameters
  - Vector normalization -> boundary crossing issues
  - particle kinematics -> energy-momentum non-conservation
  - These parameters have to be handled with care
- A study case by case of usage of float can reveal usage patterns where only some parts of the algorithms have to be executed in double
  - Speedup for vectorized algorithms can be very important
  - Examples: field map interpolation, algorithms for many geometry solids, …

Discussion…