Parallelising Initialisation of the Geant4 geometry

J. Apostolakis, CERN EP/SFT

Outline

- Context
- What does the voxelisation do?
- Prototype parallelisation
- First results
- Improvements
- Future steps

Context

- At the start of Geant4 run, the kernel initialises two key modules
  - The geometry modeller
  - The physics processes
- The geometry modeller prepares optimisation structures to speedup
  - locating points in the geometry model;
  - computing the length to the next boundary from such a location;
  - Computing distance to the closest boundary (safety – can be an underestimate)
- The initialisation time is significant for large setups
  - can be 1-few minutes for a current generation LHC detector
  - Hurts interactive use most – yet a batch job will benefit a bit from reducing it!

What does the ‘voxelisation’do?

- Creates the ‘best’ slices (in 1, 2 or 3 dim.) & a list of volumes each contains
  - Create ‘optimisation’ voxels for each volume with ‘enough’ daughter volumes
  - Tries each cartesian axis, slicing it in min(voxel_density * #volumes, upperLim)
  - Create a list of volumes in each slice, and merge equals => ‘equivalent slice’
  - If an equivalent slice has ‘enough’ daughters, iterate in another axis
  - (Re)calculate the extent of its daughter volumes
    - Restricting for 2nd axis (or 3rd) using the ‘walls’ the cuts along previous axes’ extents
- If a setup has thousands of volumes can require substantial time
  - Six ordering of axes: x/y/z, x/z/y, y/z/x, y/x/z, z/x/y, z/y/x
  - The voxel structure that ‘scores’ best is kept and used; others are deleted.

Benefits from 'optimisation' voxels

In volumes with multiple daughters, G4Navigator will benefit in every key call from the presence of 'optimisation' voxels:

- 'LocateGlobalPointAndSetup' considers only the candidate volumes in the current voxel (possibly at multiple levels),
- ComputeStep intersects only with volumes in the voxels in the starting voxel, and potentially any additional ones in the voxels along a track's path (none twice – protection exists to 'see' only those not already intersected.)
- ComputeSafety considers only the volumes in the current voxel, plus any in nearby voxels up to the 'rolling' (evolving) estimate of the safety.

Motivation for parallelisation

- In a large detector there can be hundreds or thousands of logical volumes
  - Many will have several to hundreds (or more) daughter volumes
- Voxelising each logical volume is independent of the others
  - Perfect opportunity to use multiple parallelism to reduce initialisation time!
- Every* Geant4 execution must undertake this at run startup
  - Except if you store the voxel structures
- For a large geometry the time to create the voxels can be minutes
  - It is a substantial portion of the initialisation time of Geant4
- It is our goal this year to create a first version of parallel voxelisation

Prototype - overview

- The G4GeometryManager is responsible for undertaking the ‘Optimisation’ of the geometry
  - The RunManager should call CloseGeometry() if the geometry was changed
  - In MT we can choose to postpone the voxelisation until all threads (tasks) are instantiated or started up – so they can participate in the work
- Now, as an option, the master thread
  - creates a list of logical volumes which need to have ‘optimisation’ voxels created
  - instead of actually doing the optimisation work.
- Then, when the workers (tasks or threads) are started up, they will share the work of voxelising those volumes

Prototype - what each worker does

- The WorkerRunManager calls G4GeometryManager's UndertakeOptimisation
  - ObtainVolumeToOptimise() gets a G4LogicalVolume to optimise, if any are still available (else nullptr)
- To ensure the geometry's state is fully prepare we must coordinate:
  - A barrier ensures that no task/thread can start simulation until the work of initialisation (voxelisation) has finished
- If a task started simulating early in an unvoxelised logical volume, it would
  - be slower, possibly substantially slower, as it would iterate over all candidate daughter volumes (not just the ones in the current voxel for locate, or the voxels in a track's path for compute path)
  - it could also (infrequently) get a slightly different value for safety and thus degrade reproducibility.

Implementation

- New methods
  - BuildOptimisationsParallel() is called by G4WorkerRunManager to initialise.
- Auxiliary methods
  - ReportVoxelInfo() to write out for verification.

Turning Parallel Optimisation On/Off

- To request it or turn it off you need to use *G4GeometryManager*'s method
  - RequestParallelOptimisation(optimise, verbose)
- Set 'verbose=true' to obtain statistics on the volumes with biggest contributions to memory size and CPU time for voxelisation.

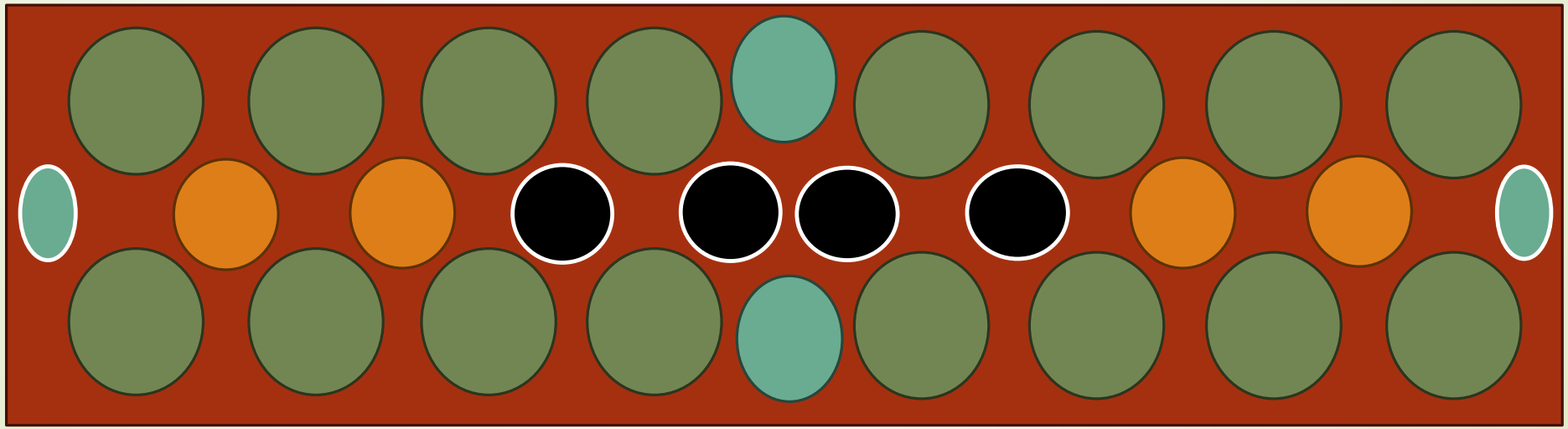
First Benchmarks

First benchmarks - revised EMEC using (old) General Trap (Max M1Max - 6+2 cores)

Workers	Real (elapsed)	User time	Speedup	Efficiency'		
1	45.7					
2	24.4	46.7	**1.9**	94		
4	12.7	46.5	**3.6**	90		
6	10.6	50.2	**4.3**	72		
8	8.6	53.2	**5.3**	66		

Limitations

- This initial version targets ‘big’ geometries
  - with a large number of ‘mother’ volumes, each with many daughters
- It is not well suited for a geometry with 1 (or few) such mother volumes
  - Only a few tasks would be created – little (or limited) speedup



Improvements – ongoing & future

- Cleanup of static methods – done by Gabriele already
- Move ‘call’ from WorkerRunManager to kernel (WorkerRunManagerKernel)
  - To (fully) enable use in tasking ‘mode’
- Further testing and benchmarking

- Future extension(s)
  - Deal with 1 (or few) big mother volumes – the volumes with the most daughters would be split into one task per ordering of axes
  - Create a class to compare voxel structures – for robustness testing
  - Separate class for parallel voxelisation (split from G4GeometryManager)

Summary

- A first version of parallel initialisation of the geometry was created
  - Released in 11.3-beta in June
  - First trials in CMS revealed some issues with its tasking (with no master…)
- Improvements are ongoing
  - To address better integration with tasking and
  - future parallelisation of initialisation
- First benchmarks are promising for realistic HEP geometries

Some details of voxelisation

- What does initialisation of the geometry of a large geometry involve?
  - Create ‘optimisation’ voxels for each volume with ‘enough’ daughter volumes
  - Try each cartesian axis, slicing it in min(voxel_density * #volumes, upperLim)
  - Create a list of volumes in each slice, and merge equal entries => equivalent
  - If a slice along current axis has ‘enough’ daughters, iterate in another axis
- If a setup has thousands of volumes can require substantial time
  - Six orders for the candidates
  - (Re)calculate the extent of its daughter volumes
    - Initially along that axis (x, y or z)
    - For 2nd axis (or 3rd) the extent while ‘sliced’ along previous axes’ extents