Profiling study of Geant4-GATE VRTs discussion

Nicolas Karakatsanis
Alexander Howard

Geant4 Users Workshop Medical/GATE Parallel Session
13th September 2007, Hebden Bridge, UK

Contents

- Introduction
- Simulation efficiency with voxelized phantoms
- Compressed Voxel Parameterization
- Profiling study - Conclusions
- Geometry VRTs so far
- Discussion of possible new geometry VRTs
- Suggested actions

Introduction

- GATE performance degrades when voxelized maps (attenuation or emission distributions) are used
  - Simulation times become prohibitive for clinical or pre-clinical simulation studies
- Profiling studies indicate a performance bottleneck on the Geant4 navigator function
  - Collaboration between G4 and GATE developers to optimize the efficiency

Simulation efficiency with voxelized phantoms

- Geant4-GATE features (our analysis)
  - Deal with each voxel as a different material region
  - At least one step is forced in each material region boundary
  - Therefore
    - G4 stepping implementation can be one of most important (if not the most important) factors that affects simulation speed
      - when using large voxelized phantoms
- Possible solution
  - If the material in one voxel is the same as that in its neighbouring voxel,
    - Treat those two voxels as one region (no region boundary between those two voxels)
  - Check that condition for all voxels

Simulation efficiency with voxelized phantoms

- Possible solution (..continue)
  - Create an option in stepping function
    - to test if the material in next region is the same as current one.
  - If yes
    - no boundary is applied here and
    - a normal step is taken
  - A flag can turn on/off the previous option

Compressed Voxel Parameterization

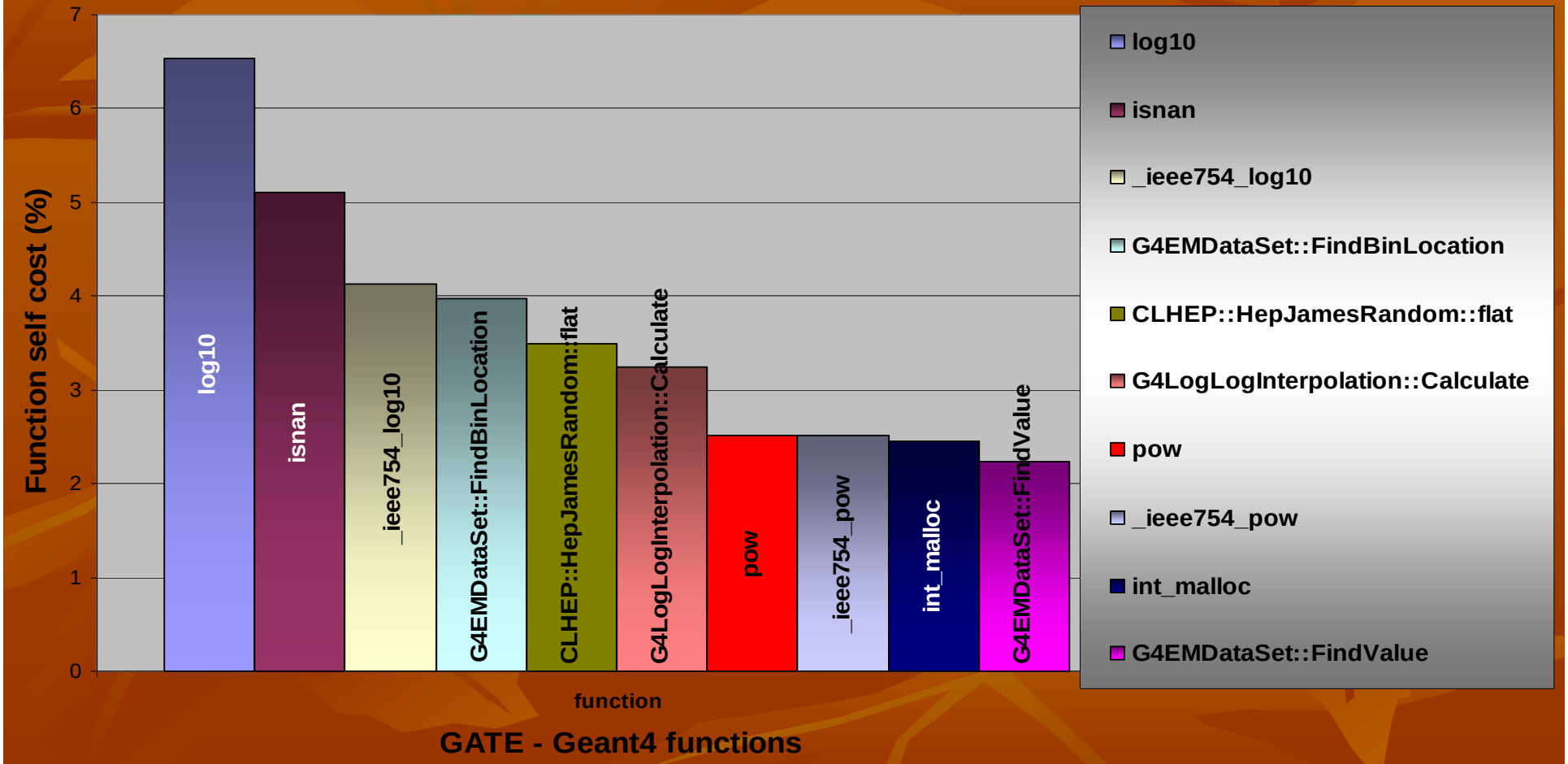
- Solution developed in GATE
  - Application of a “compressed” voxel parameterization
    - generate a phantom where voxel size is variable
    - All adjacent voxels of the same material are fused together to form the largest possible rectangular voxel.
  - Compressed parameterization uses
    - less memory and also
    - less CPU cycles
  - It is possible to exclude regions in the phantom from being compressed through the use of an "exclude list" of materials
- Discussion
  - Initial purpose of implementation
    - less memory usage – larger phantoms can be simulated
  - Side-effects
    - less CPU cycles usage - reduce the simulation time but only by maybe 30%
  - Need for a better speed-up factor
    - Investigation of a more efficient way to track particles in a voxelized geometry
- Contact: Richard Taschereau: RTaschereau@mednet.ucla.edu

Profiling studies

- Profiling studies using GNU tools
  - Callgrind
    - Simulation set-up
      - Biograph6 scanner
      - 1 sec acquisition time
      - Activity level -> 10kBq
  - Compare flat profiles and call graphs of following cases
    - NEMA cylindrical analytical phantom
    - NCAT thorax voxelized phantom (using default voxel parameterization)
    - NCAT thorax voxelized phantom (using compressed voxel parameterization)

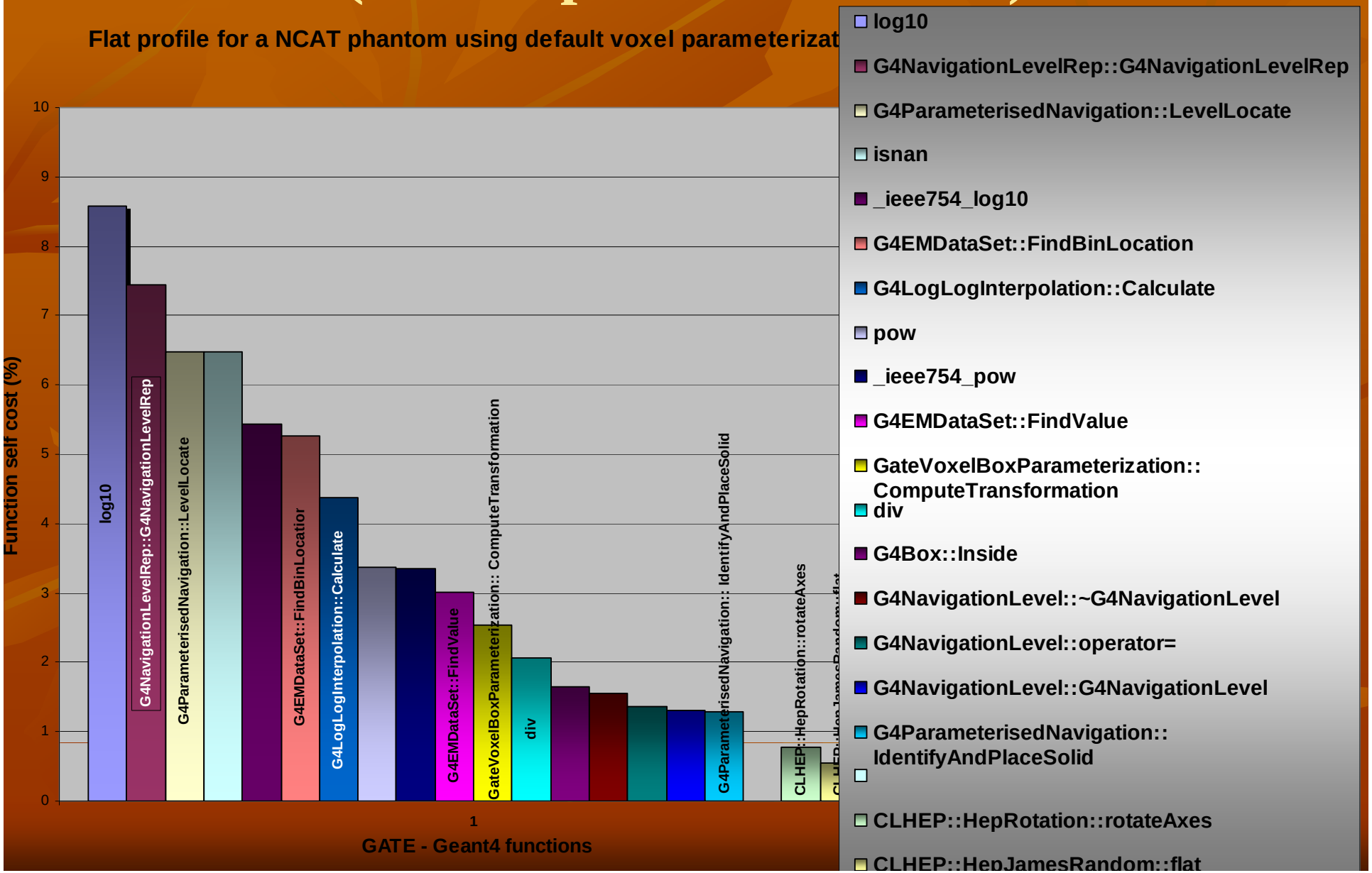
Callgrind study analytical NEMA phantom

Flat profile for an analytical cylindrical NEMA phantom

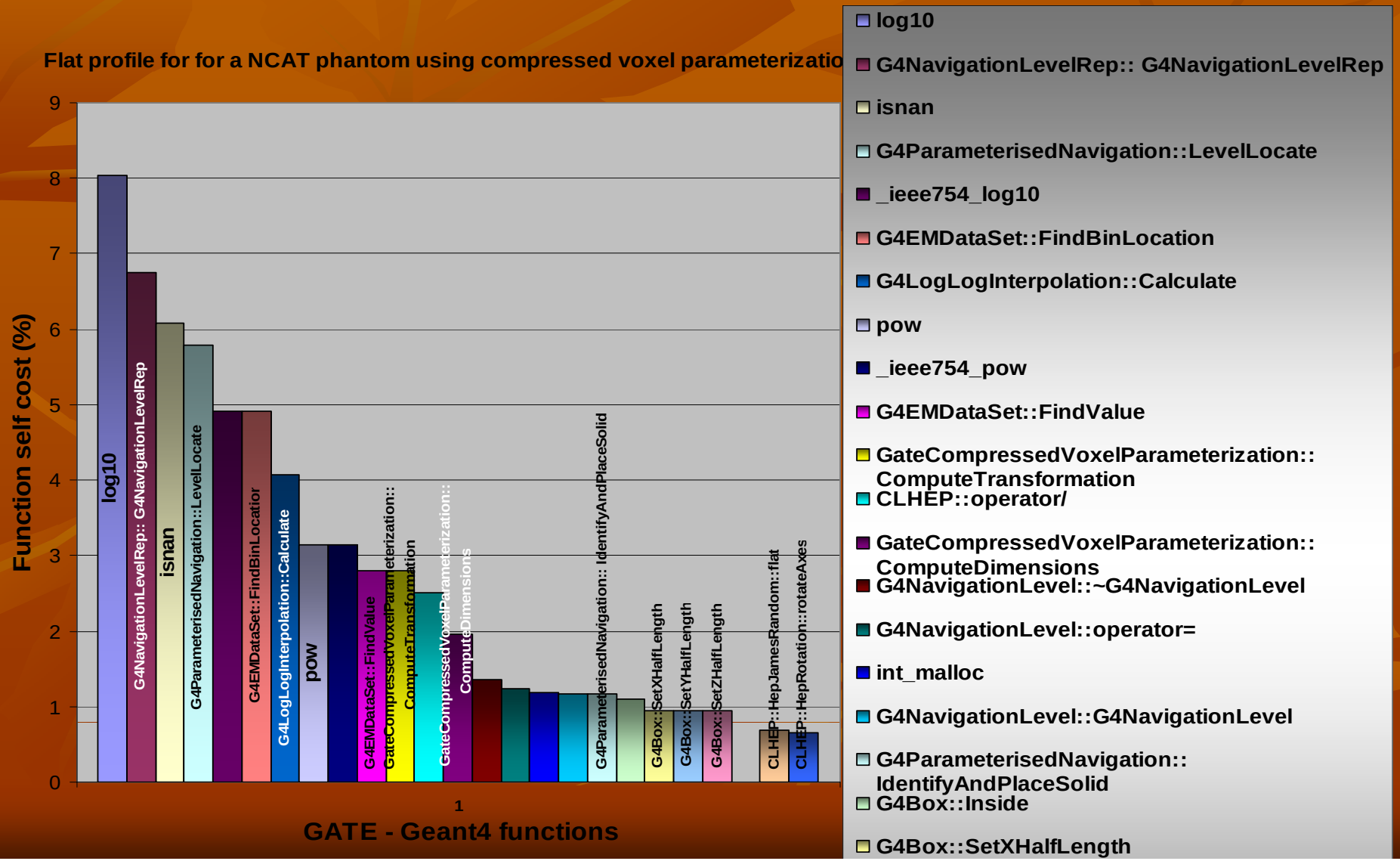


GATE - Geant4 functions

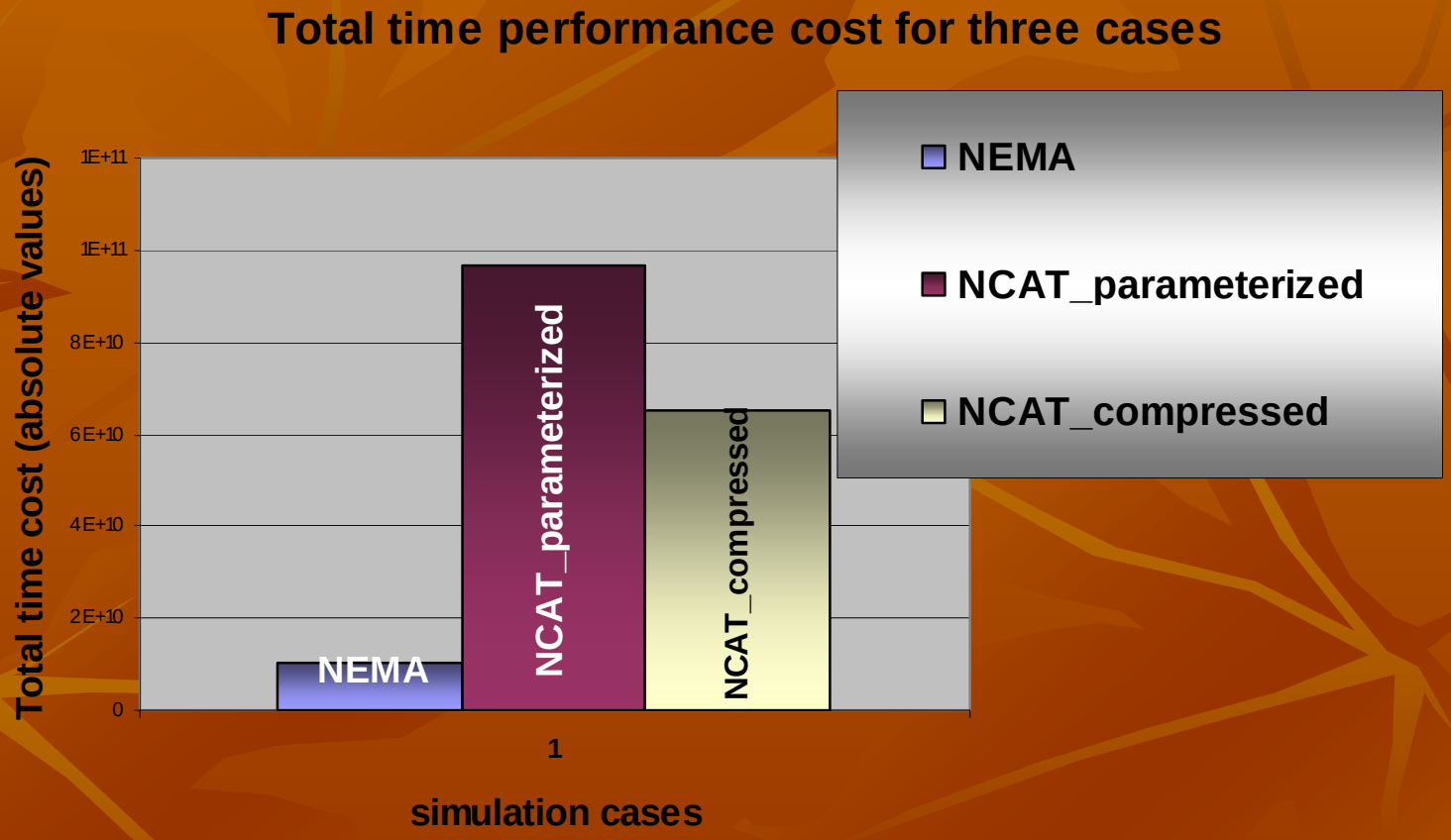
Callgrind study voxelized NCAT phantom (default parameterization)



Callgrind study voxelized NCAT phantom (compressed parameterization)

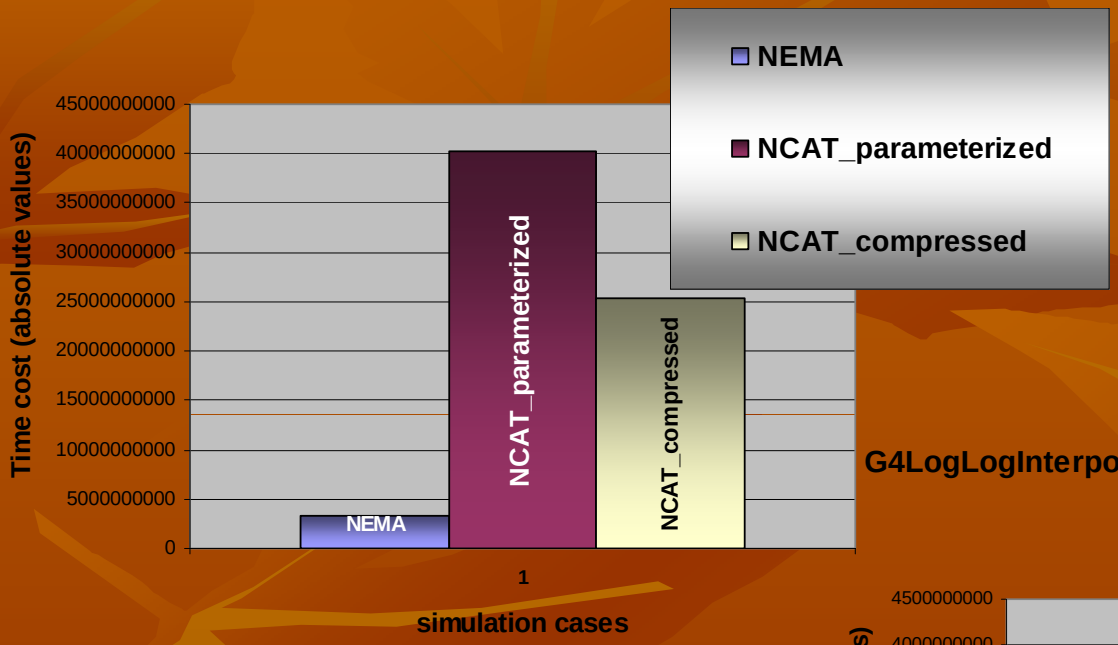


Callgrind study
Comparison of time-performance cost



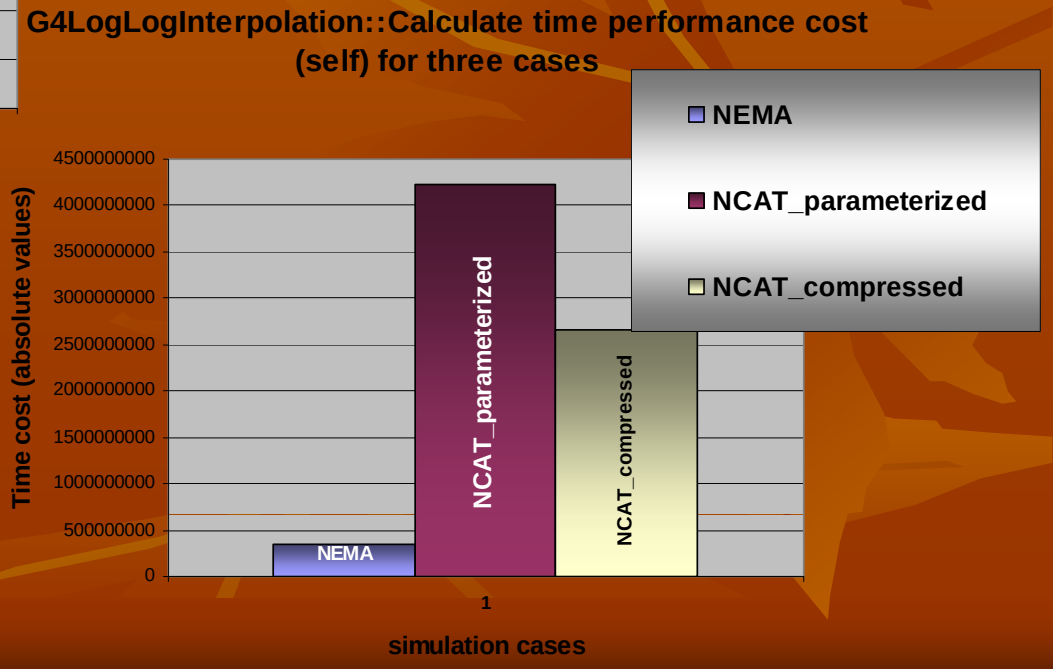
Total time cost refers to the total time required for the execution of the simulation
Absolute cost values are presented with arbitrary units
The total cost increases by 814% and 517% when a voxelized phantom and a compressed phantom is used respectively

Callgrind study
Comparison of time-performance cost



G4EMDataSet::FindValue exhibits one of the highest time inclusive costs (absolute values) among the G4functions which are called by the G4SteppingManager

Inclusive time cost is the time spent on the function itself and all of its callees

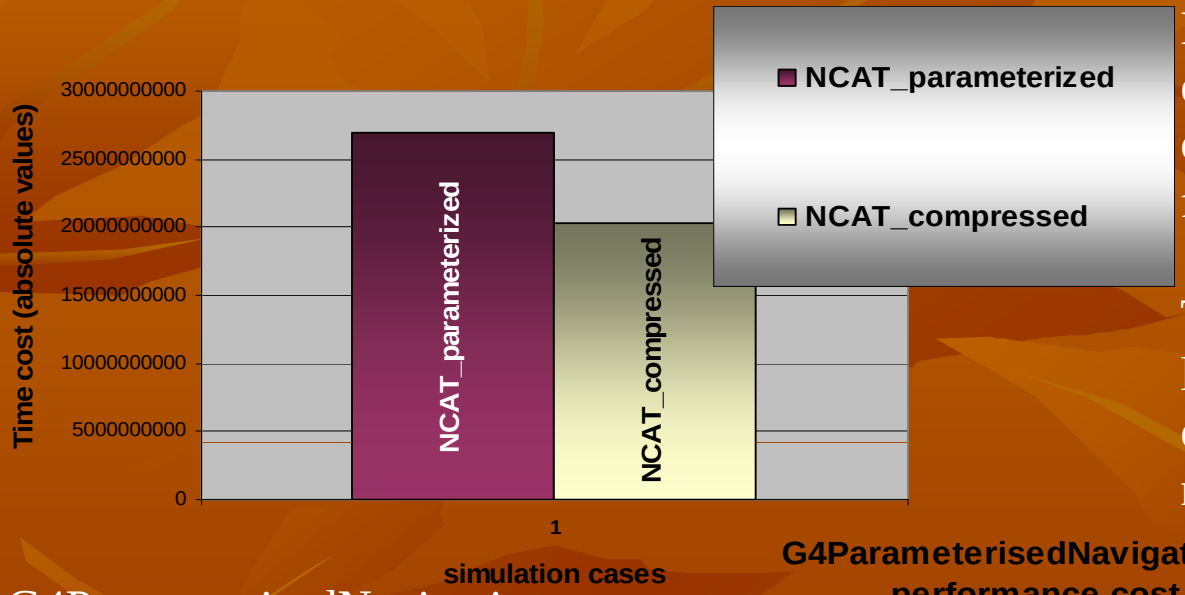


G4LogLogInterpolation::Calculate exhibits one of the highest time self costs (absolute values) among the G4functions which are called by the G4EMDataSet::FindValue

Self time cost is the time spent only on the function itself

Callgrind study
Comparison between default and compressed parameterization

G4ParameterisedNavigation::LevelLocate time performance cost (inclusive) for three cases



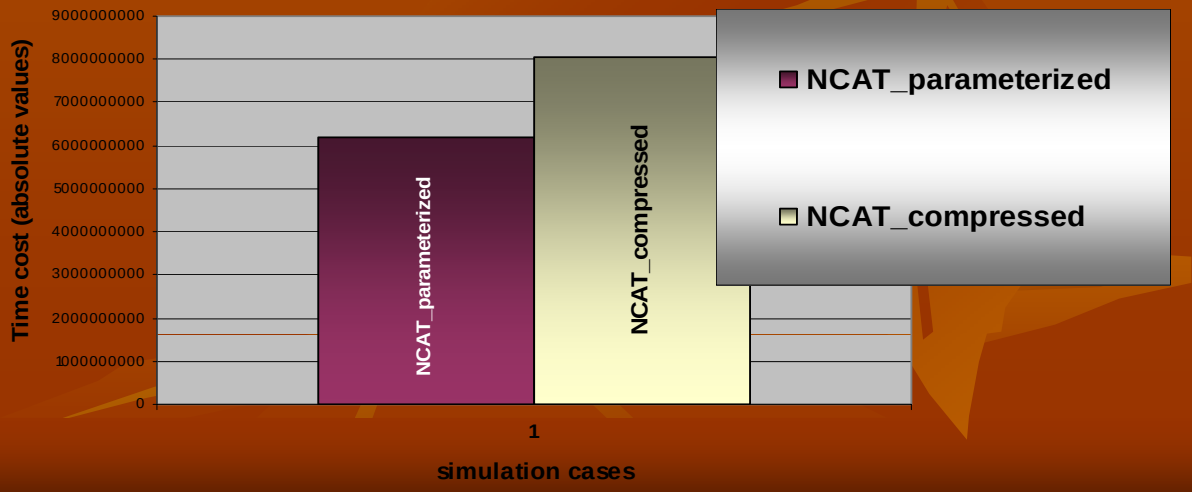
G4ParameterisedNavigation:: LevelLocate exhibits one of the highest time inclusive costs (absolute values) among the functions of the G4Navigator class

This function is located at libG4navigator.so and calls most of the functions used by the navigator

G4ParameterisedNavigation:: IdentifyAndPlaceSolid determines the parameterization type applied for each voxel

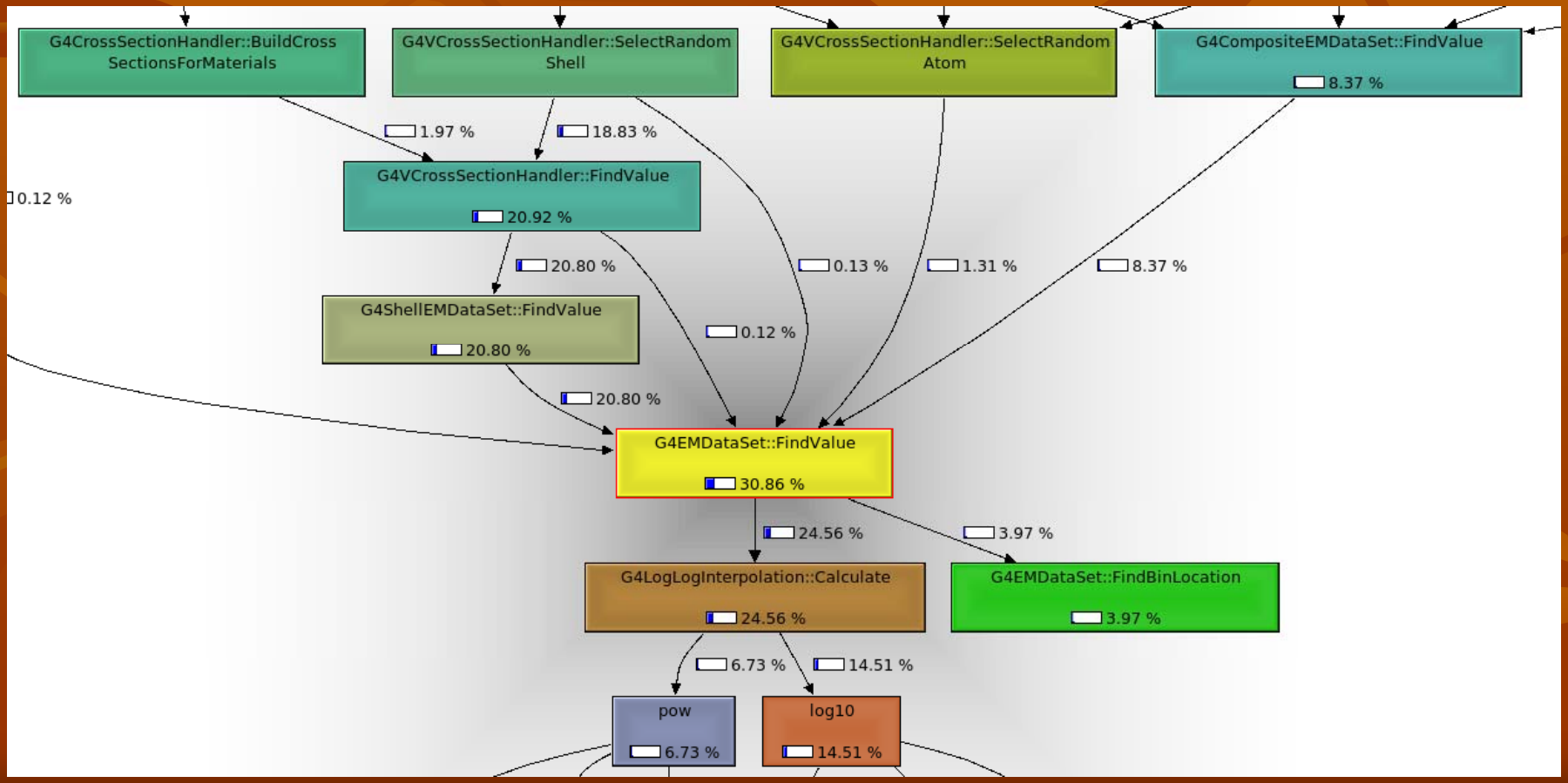
Compressed voxel implementation adds up the inclusive time cost of this function, because it adjusts the voxel dimensions depending on its material attribute

G4ParameterisedNavigation:: IdentifyAndPlaceSolid time performance cost (inclusive) for three cases



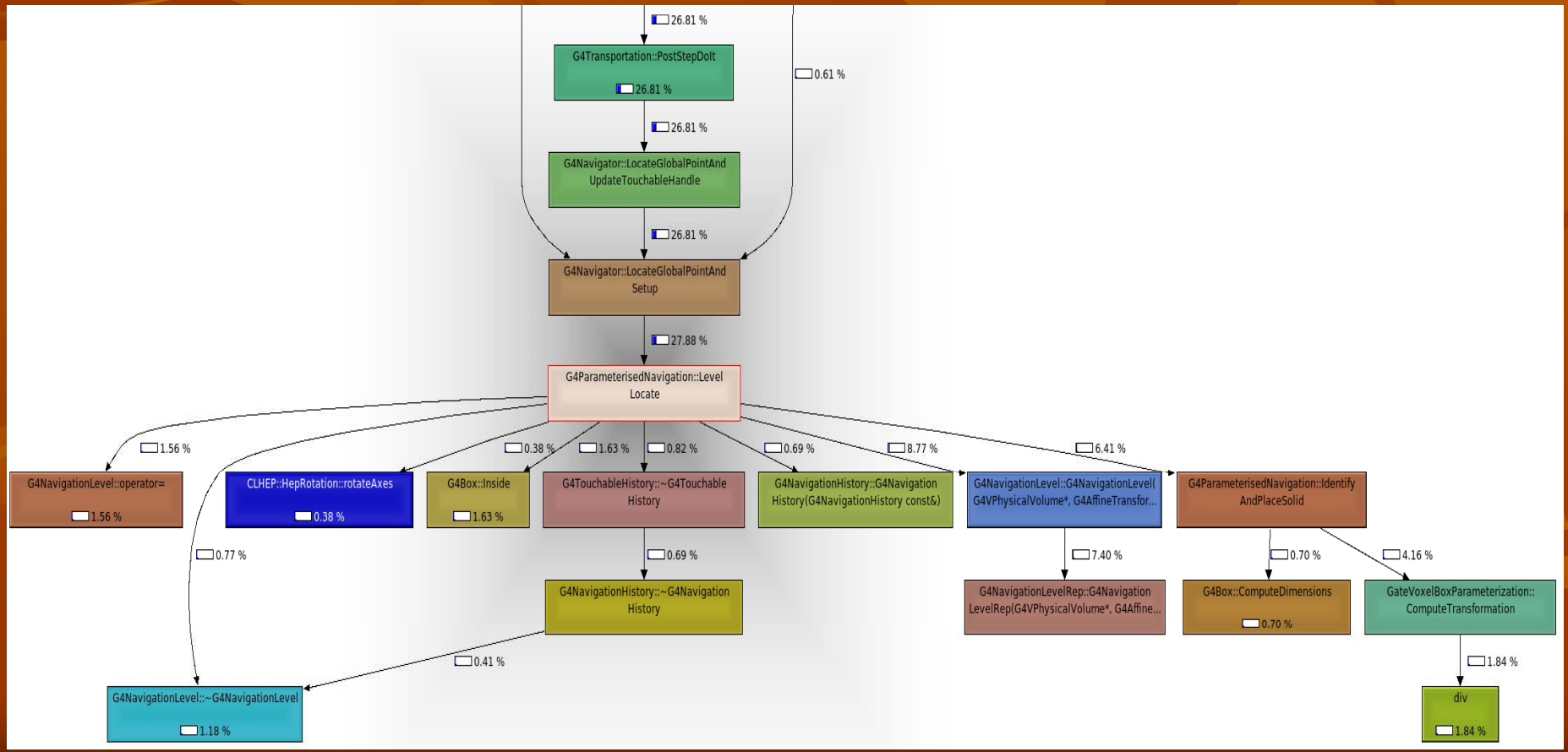
Callgrind study - Call graph NEMA analytical phantom

G4EMDataSet::FindValue function spends 30.86% of the total cost by completing it's calls to callees functions and by returning calls made to this function from other caller functions.



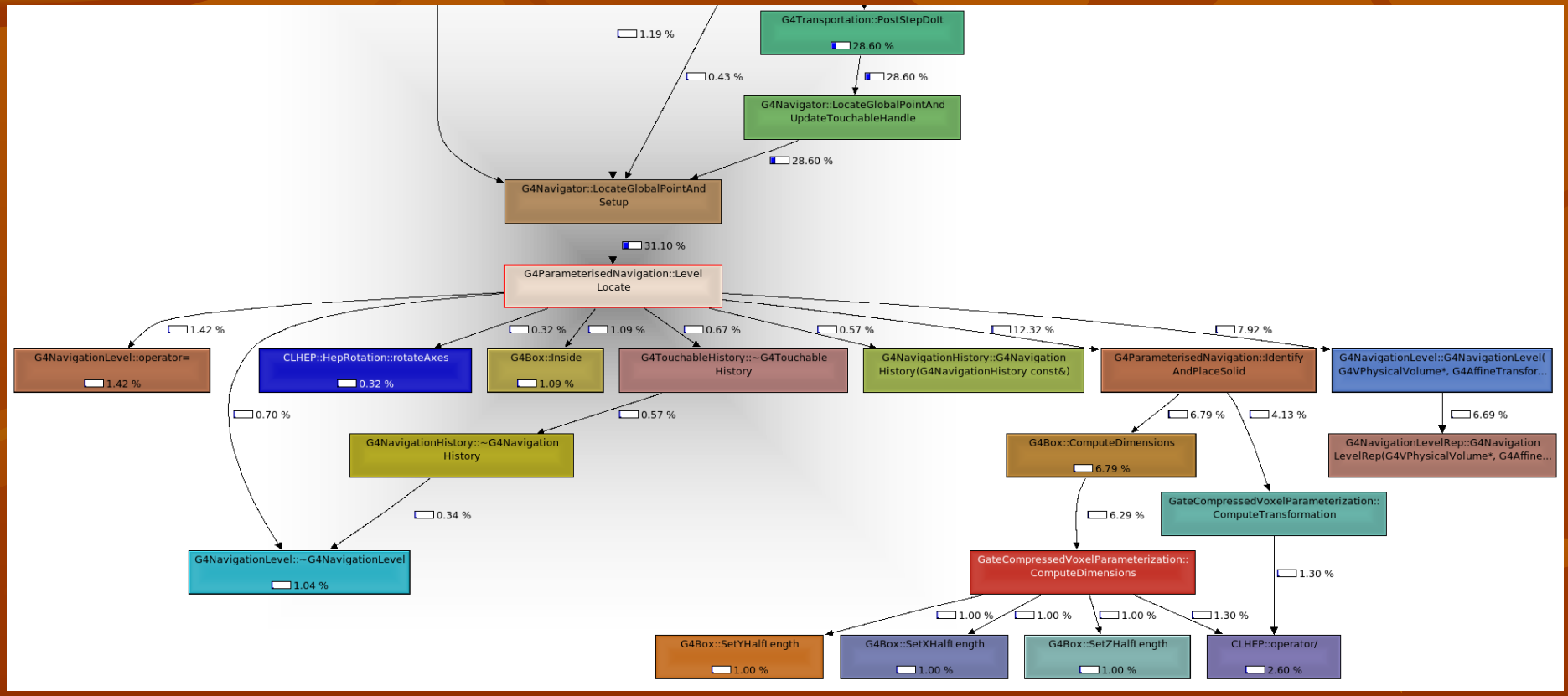
Callgrind study - Call graph
NCAT voxelized phantom (default parameterization)

G4ParameterisedNavigation::LevelLocate function spends 27.88% of the total cost by by completing it's calls to callees functions and by returning calls made to this function from other caller functions.



Callgrind study - Call graph
NCAT voxelized phantom (compressed parameterization)

G4ParameterisedNavigation::LevelLocate function spends 31.10% of the total cost by by completing it's calls to callees functions and by returning calls made to this function from other caller functions.



Profiling study conclusions

- In the case of voxelized phantoms
  - Significant degradation of the G4-GATE performance
    - 8 times slower with default parameterization
    - 5 times slower with compressed parameterization
  - G4NavigationLevelRep::G4NavigationLevelRep exhibits the highest self time cost (max. 7.44%)
  - G4ParameterisedNavigation::LevelLocate exhibits the highest inclusive time cost among functions called by the G4steppingManager
  - Compressed voxel parameterization achieves speed-up of 30-40% only
  - Need for more efficient version of stepping manager and navigator optimized for G4 medical applications
- In the case of analytical phantoms
  - G4EMDataSet::FindValue (located at libG4emlowenergy.so) appears to cause the most important time cost
    - Primer caller of the G4 function with the highest combination of self and inclusive time cost (G4LogLogInterpolation::Calculate) in the case of analytical phantoms
  - Discussion between G4 and GATE developer communities
    - Guidance on which EM options used for different medical applications

Event Biasing in Geant4

PHYSICS BASED BIASING:

- Built in biasing options
  - Primary particle biasing
  - Radioactive decay biasing
  - General hadronic leading particle biasing
  - Hadronic cross section biasing
- User defined biasing
  - G4WrapperProcess
- See release documentation for detailed instructions on how to implement the various biasing techniques

GEOMETRICAL BIASING:

- Importance Sampling
- Weight Window

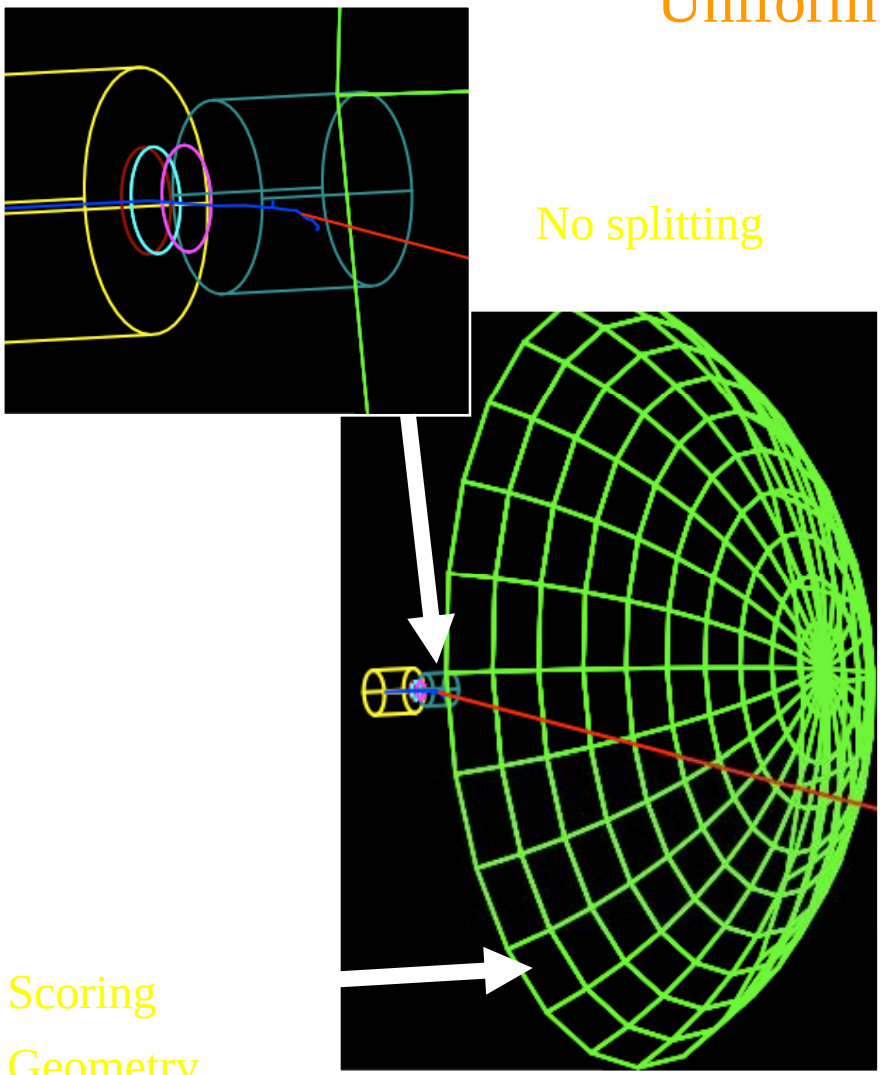
Primary Particle Biasing

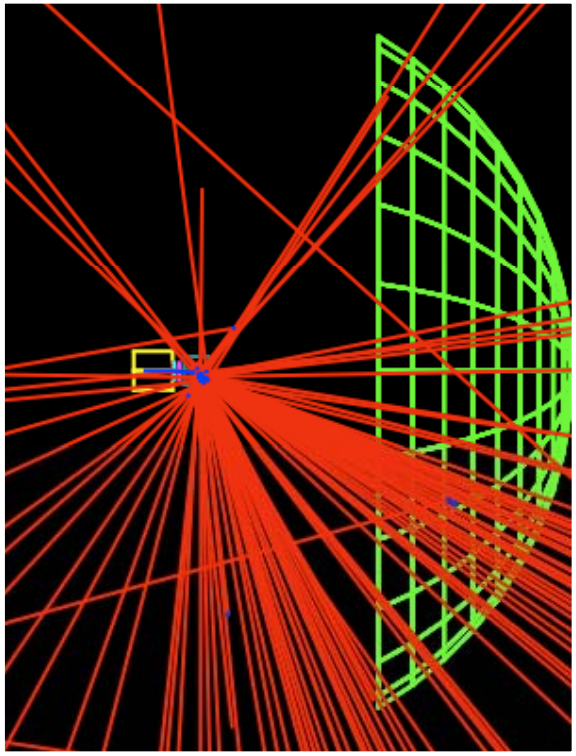
- **Use case:**
  - Increase number of high energy particles in cosmic ray spectrum
- **Increase number of primary particles generated in a particular phase space region of interest**
  - Weight of primary particle modified as appropriate
- **General implementation provided by G4GeneralParticleSource class**
  - Bias position, angular and energy distributions

User Defined Biasing: G4WrapperProcess

- **Implement user defined biasing through G4WrapperProcess**
  - **A process itself, I.e, inherits from G4VProcess**
  - **Wraps an existing process**
    - **By default, function calls are forwarded to existing process**
  - **Non-invasive way to manipulate the behaviour of a process**
- **To use:**
  - **Subclass G4WrapperProcess and override appropriate methods, e.g, PostStepDoIt**
  - **Register sublcass with process manager in place of existing process**
  - **Register existing process with G4WrapperProcess**

Uniform Bremsstrahlung Splitting





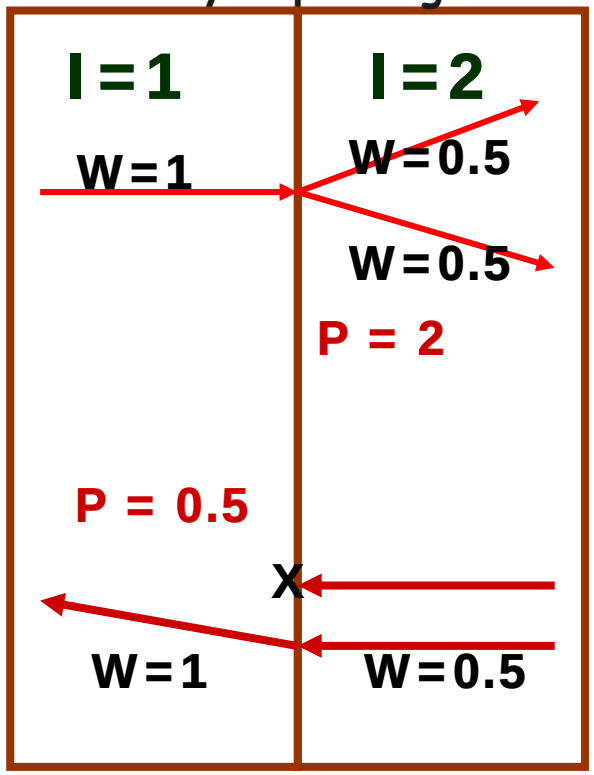
Geometric Biasing

The purpose of geometry based event biasing is to save computing time by sampling less often the particle histories entering "less important" geometry regions, and more often in more "important" regions.

* Importance sampling technique
* Weight window technique

Importance sampling technique

- Importance sampling acts on particles crossing boundaries between "importance cells".
- The action taken depends on the importance value assigned to the cell.
- In general, a track is either split or plays Russian roulette at the geometrical boundary depending on the importance value assigned to the cell.



- Survival probability (P) is defined by the ratio of importance value.
  $P = I_{post} / I_{pre}$
- The track weight is changed to W/P.

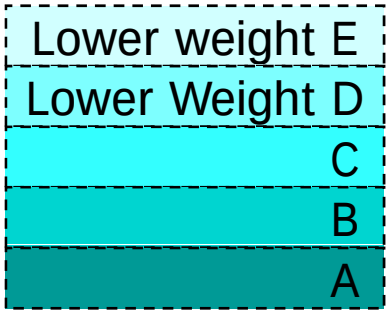
- Splitting a track (P > 1)
  - E.g. creating two particles with half the 'weight' if it moves into volume with double importance value.

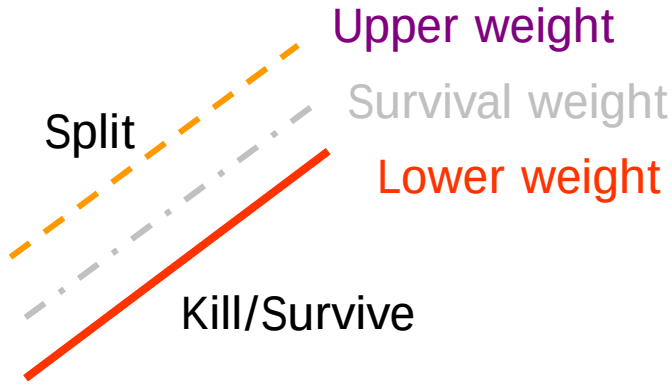
- Russian-roulette (P < 1) in opposite direction
  - E.g. Kill particles according to the survival probability (1 - P).

The Weight Window Technique

- The weight window technique is a weight-based algorithm – generally used together with other techniques as an alternative to importance sampling:
  - It applies splitting and Russian roulette depending on space (cells) and energy
  - User defines weight windows in contrast to defining importance values as in importance sampling

- A weight window may be specified for every cell and for several energy regions: *space-energy cell* .

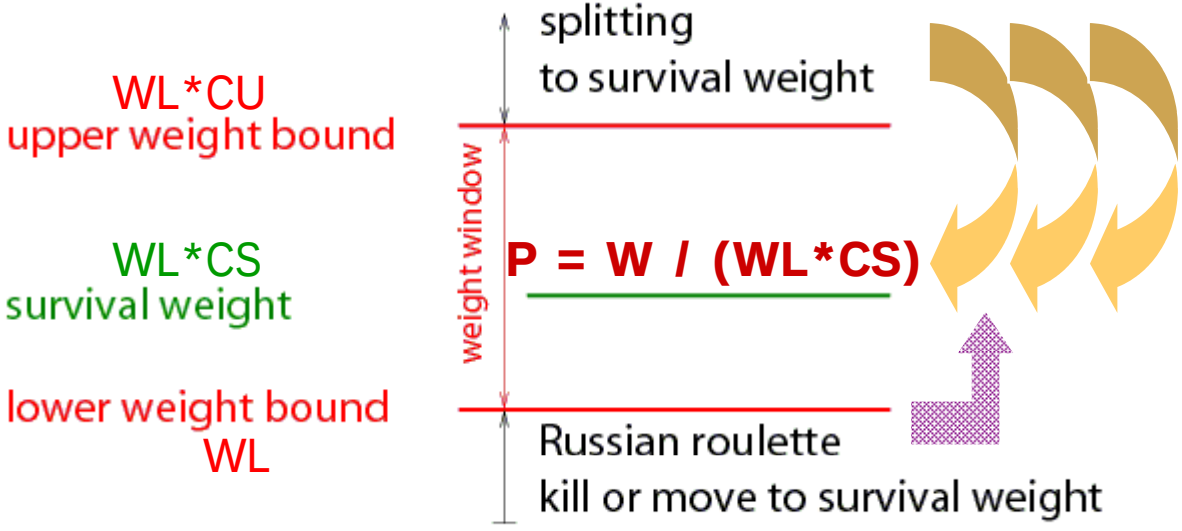




- Apply in combination with other techniques such as cross-section biasing, leading particle and implicit capture, or combinations of these.

The weight window technique (continue)

- Checks the particle weight
  - Compare the particle weight with a 'window' of weights defined for the current *energy-space* cell
  - Play splitting or roulette in case if it is outside, resulting in 0 or more particles 'inside' the window
    - E.g. WL is a lower weight bound of a cell.
      CU and CS are upper limit factor and survival factor, respectively.
      - W > WL*CU Split track
      - W < WL*WL Roulette



Coupled Transportation

- Since release 8.2 coupled transportation has been included
- This is a generic form of parallel navigation
- Geometrical biasing was migrated/copied to this formalism
  - No longer duplicate mass and parallel classes/samplers
  - Switch between mass and parallel world is through transportation assignation
  - Examples also migrated – release 9.0
- Examples also run for charged particles (and should handle magnetic fields)

Scoring/Tallies – brief overview

Tallies	MCNPX	GEANT4	FLUKA	MARS	PHITS
Standard					
Flux					
Volume	Yes	Yes	Yes	Yes	Yes
Surface	Yes	Yes	Yes	Yes	Yes
Point/ring	Yes	No	No	Yes (neutrons)	No
Current	Yes	Yes	Yes	Yes	Yes
Charge	Yes	Yes	Yes	Yes	Yes
Kinetic energy	Yes	Yes	Yes	Yes	Yes
Particle density	Yes	Yes	No	No	No
Reaction rates	Yes	No	Star (inelastic)	Yes	Yes
Energy dep.	Yes	Yes	Yes	Yes	Yes
Rapidity	No	Yes	Yes	Yes.	No
DPA	Some	??	Yes	??	??
Momentum	No	Yes	Yes	Yes	No
Pulse-height	Yes	User input	Yes	No	Yes
Termination	Partial	??	??	??	Yes
Modifiers	10	3 >11	2	2	2
Special					
Mesh	rec, cyl, sph	Arbitrary	rec, cyl	rec, cyl, sph	rec,cyl
Coincidence	Yes	(USER)	Yes	Yes	Yes
Residuals	Yes	No	Yes	Yes	Yes
Event logs	Yes	Yes	Yes	Yes	Yes
Convergence Tests	10	Error >1	Error	Error	Error

Review of Monte Carlo All-Particle Transport Codes and ..
G. W. McKinney (*Los Alamos National Laboratory*) et al,
FNDA 2006, Cape Town, South Africa, April 3-6, 2006

9

- Physics biasing
  - Existing physics based biasing fragmented
  - Identify missing biasing methods & variations between methods in other Monte Carlo codes
    - Implicit capture
    - General cross section biasing
    - Interaction forcing
    - Path length biasing
    - Advanced bremsstrahlung splitting
    - Leading particle biasing
  - Look at developing dedicated framework to provide general physics biasing in analogy with geometrical biasing
    - Manipulating physics processes/lists

VRTs for geometry so far

- Variance Reduction Techniques (VRTs)
  - Geometrical Event Biasing
    - Importance sampling
      - Photon splitting
      - Russian roulette
    - Weight window and energy sampling
- A number of VRT implementations are already available in Geant4, but
  - Some of the Geant4 classes can be customized more
    - Make efficient use of the toolikit architecture of G4
      - Design of a special navigator customized for voxel geometries
  - Further classes implementing VRTs could be developed
    - Improve of the geometry definition transported from GATE to G4

Discussion of possible new geometry VRTs

- Alternative photon splitting technique for G4
  - If the first photon of the annihilation pair is detected
    - => second photon splits into multiple photons with equal weights and total weight sum of 1
  - Degree of splitting depends on the probability of detection
    - Probability is dependent on axial position and emission angle
      - Therefore geometrical importance sampling is based on axial position and emission angle
        - Therefore prior knowledge of the geometry of the detector systems is required
    - High detection probability => photon splits into a small number of secondary photons
    - Low detection probability => photon splits into a large number of secondary photons
  - Speed up of the technique (applied in published MCs for scatter correction –*Holdsworth et al*)
    - 3 – 4 times increase in efficiency
- Additional photon transport algorithms could be implemented
  - Delta scattering including energy discrimination
- Flags implementation
  - user can easily activate or not the various VRT options depending on the requirements of its application

Suggested actions

- Geant4/GATE collaboration targets
  - Guidance on which EM parameters should be used for different medical applications (J. Perl's suggestion)
    - physics processes, range cuts, step sizes
  - Profiling using these guidances and suitable initialization
  - Use of profiling results for determination of the G4 classes need to be optimized
    - Customization of G4Navigator for voxel geometries
  - Validation of the efficiency gain achieved
    - is the speed/accuracy trade-off linear?
  - Publication of the new customized libraries