Profiling study of Geant4-GATE Discussion of suggested VRTs

Nicolas Karakatsanis, George Loudos, Arion Chatziioannou

Geant4-GATE technical meeting
12th September 2007, Hebden Bridge, UK

Contents

- Introduction
- Simulation efficiency with voxelized phantoms
- Compressed Voxel Parameterization
- Axial emission angle effect on efficiency
- Profiling study - Conclusions
- Discussion of possible new VRTs applied
- 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
  - Only 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

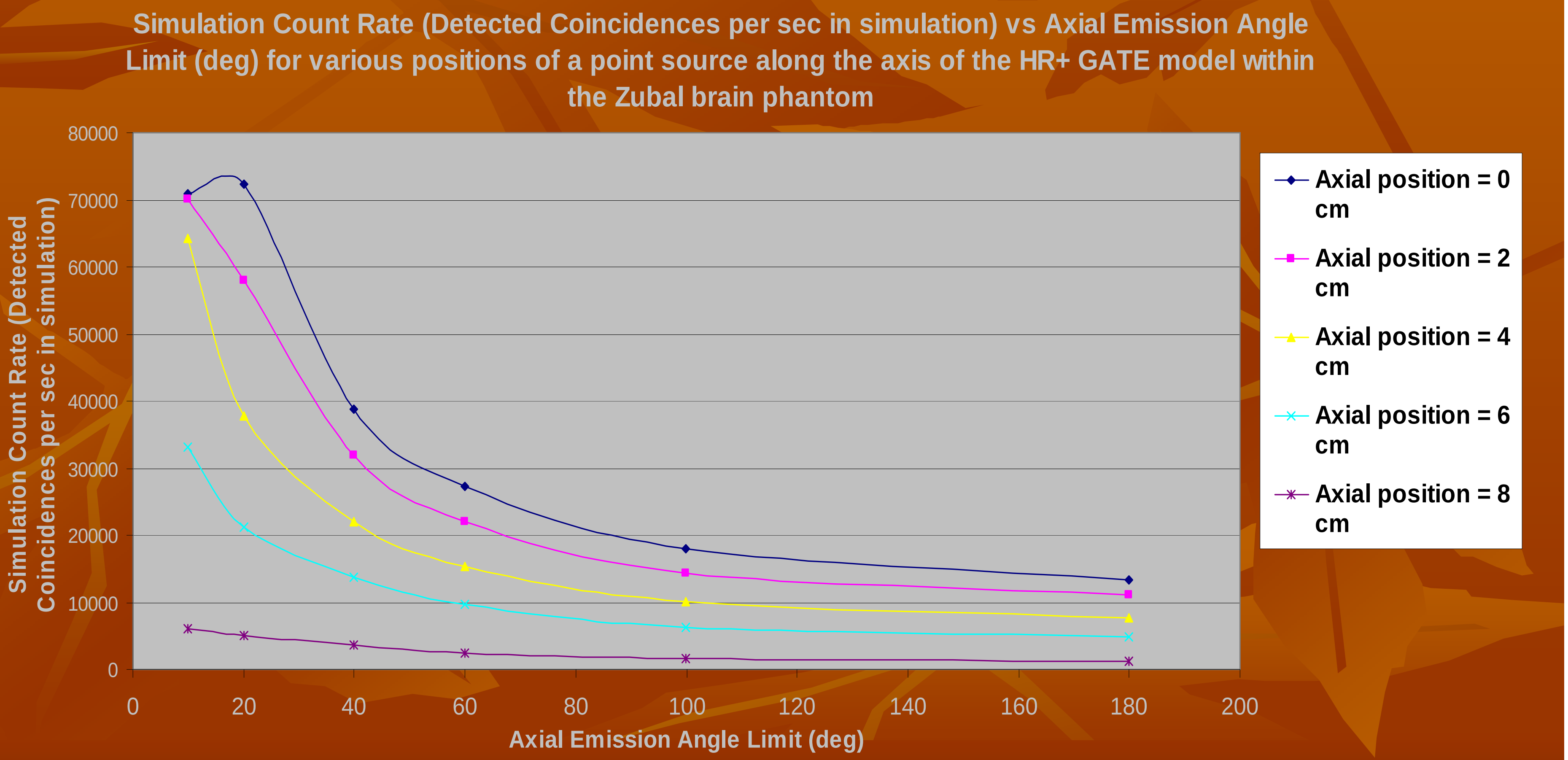
Compressed Voxel Parameterization

- 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

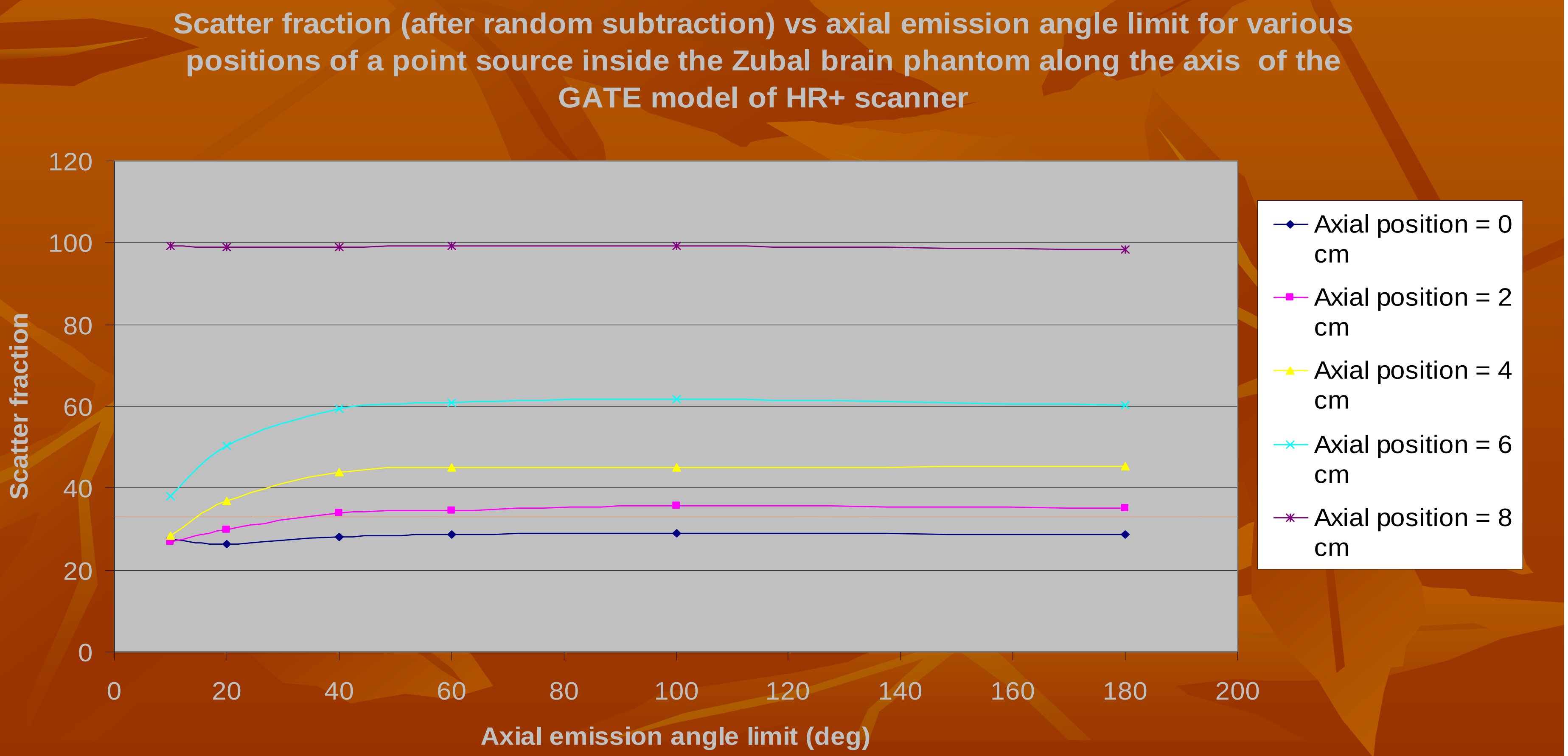
Axial emission angle effect on efficiency

A certain axial emission angle (for back-to-back emission source) optimizes the simulation count rate without significant inaccuracy in the scatter fraction



Axial emission angle effect on efficiency

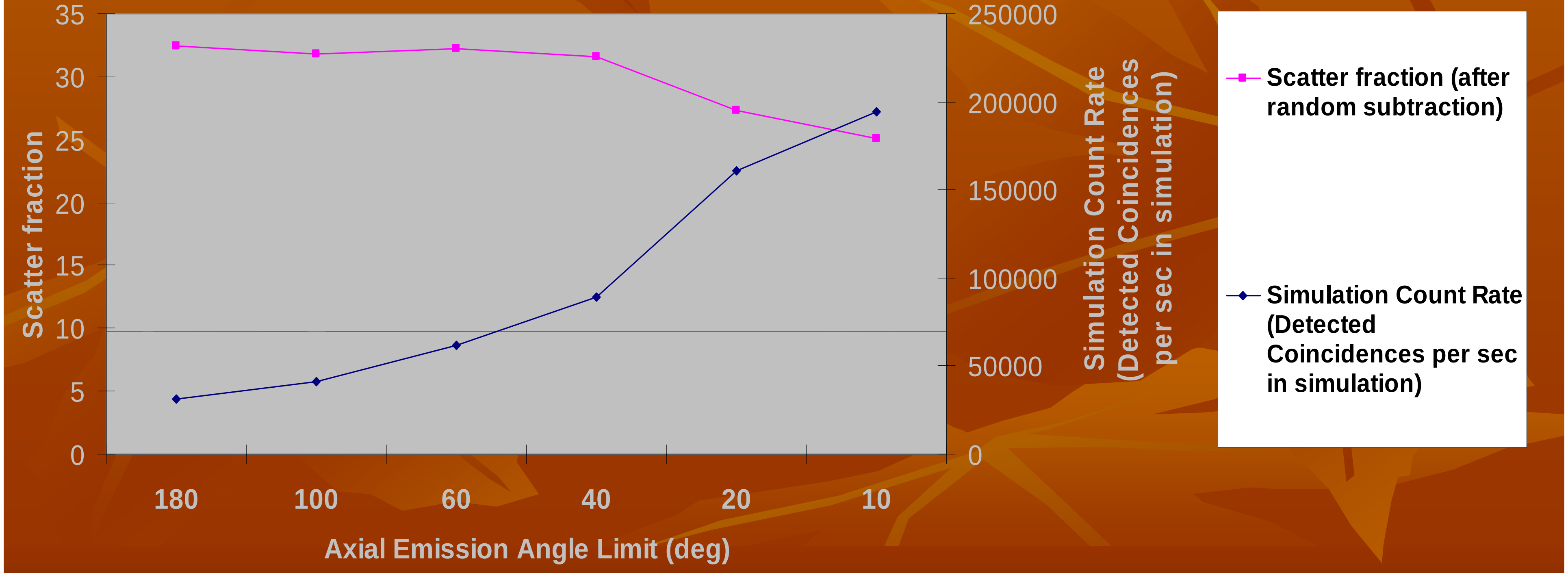
A certain axial emission angle (for back-to-back emission source) optimizes the simulation count rate without significant inaccuracy in the scatter fraction



Axial emission angle effect on efficiency

A certain axial emission angle (for back-to-back emission source) optimizes the simulation count rate without significant inaccuracy in the scatter fraction

Comparative plot of the relationship of simulation count rate and scatter fraction as a function of axial emission angle limit for a realistic activity distribution (CBF and Brain Matter) inside the Zubal brain phantom using the GATE HR+ model scanner

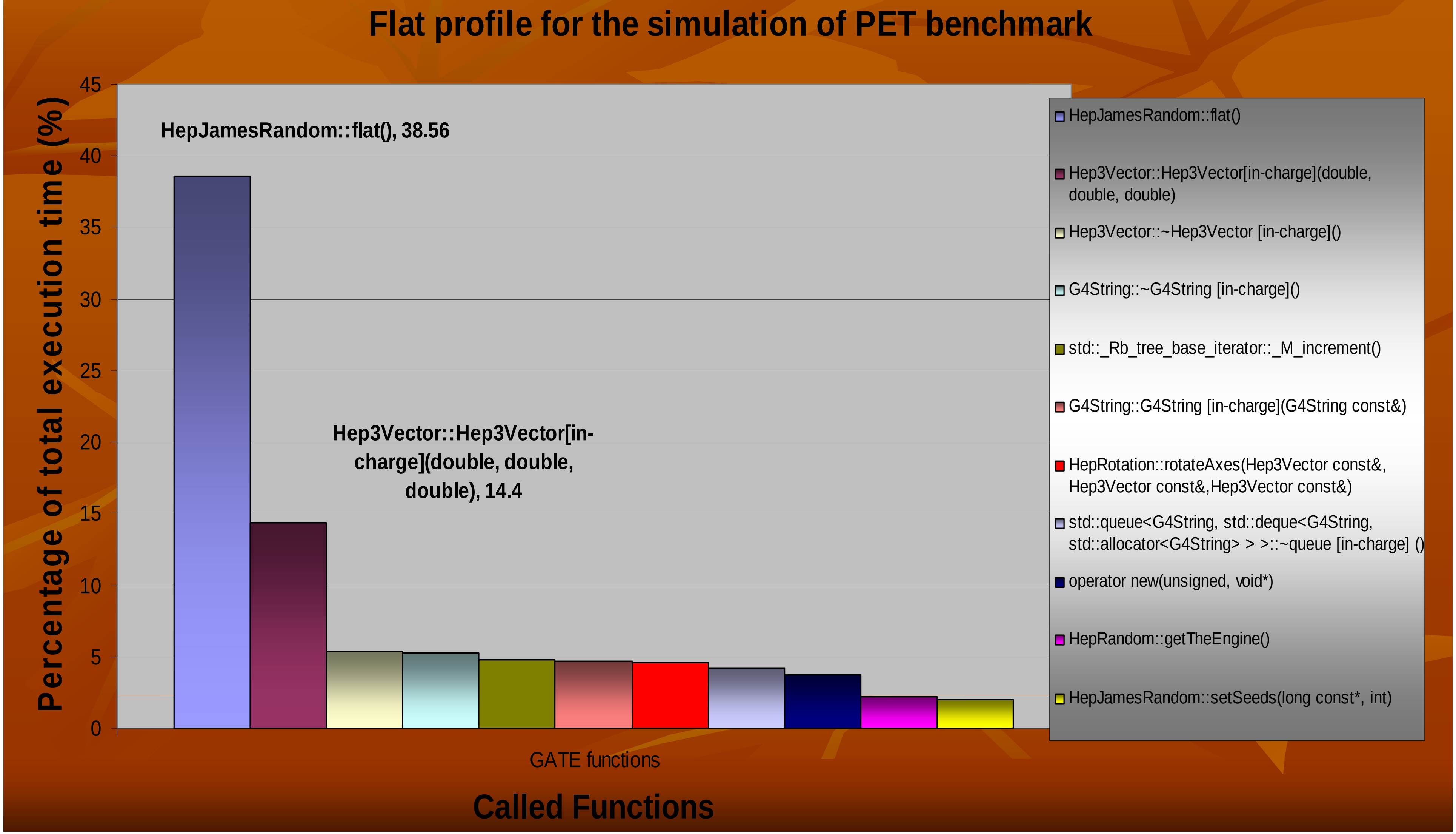


Profiling studies

- Profiling studies using GNU tools
  - Gprof : Compare flat profiles of following
    - PET benchmark
      - 10sec acquisition time
    - HR+ ECAT 962 model with Zubal high-resolution brain phantom
      - 10sec acquisition time
      - Full attenuation map (128 slices of 256x256 pixels each)
      - Simple emission map (21 central slices)
  - Callgrind: 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)

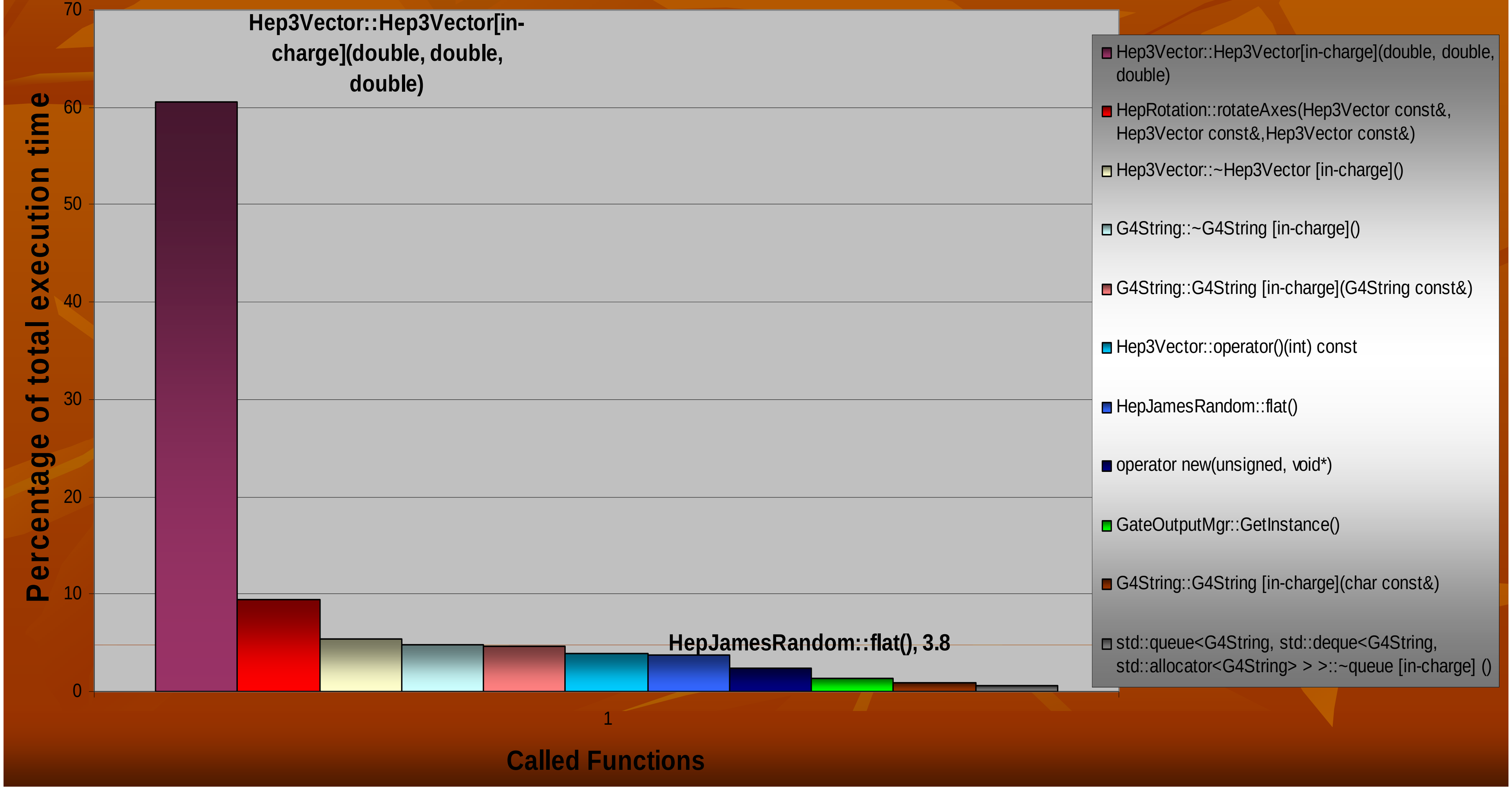
*using in all cases the GATE model of Biograph6 scanner and acquiring for 1 sec with the same activity

Gprof study – analytical phantom

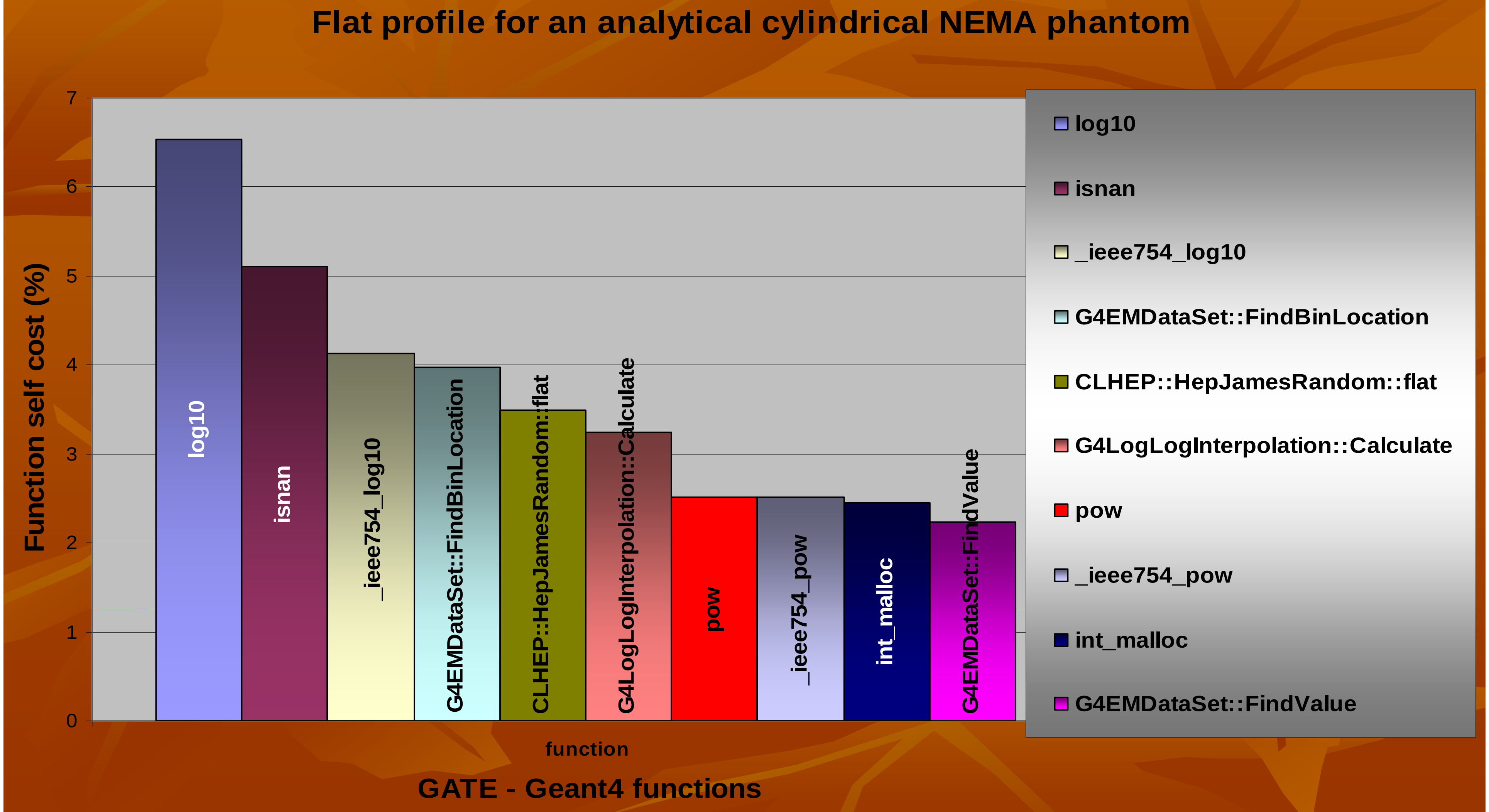


Gprof study – voxelized phantom

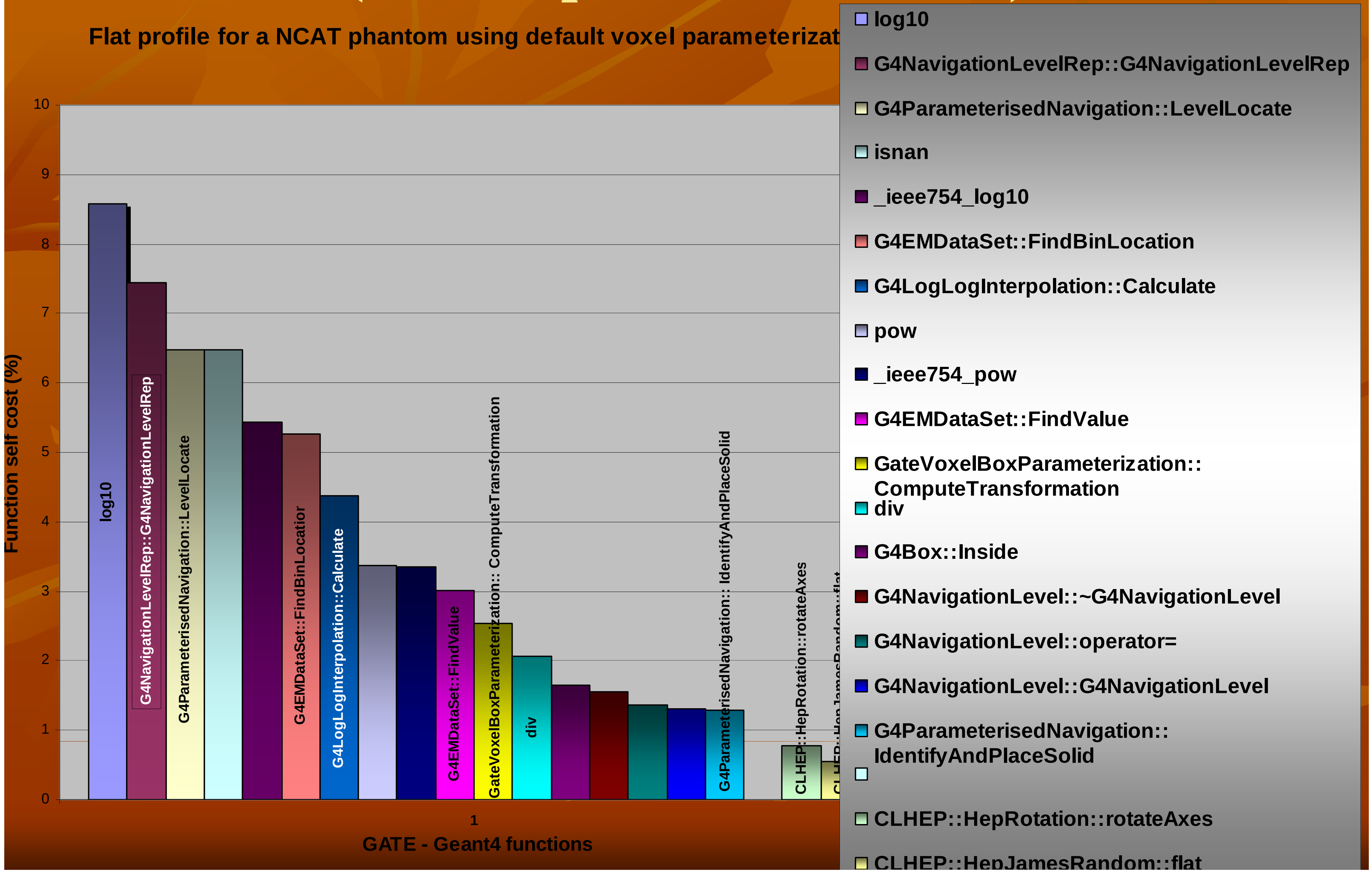
Flat profile for the simulation of HR+ with zubal phantom



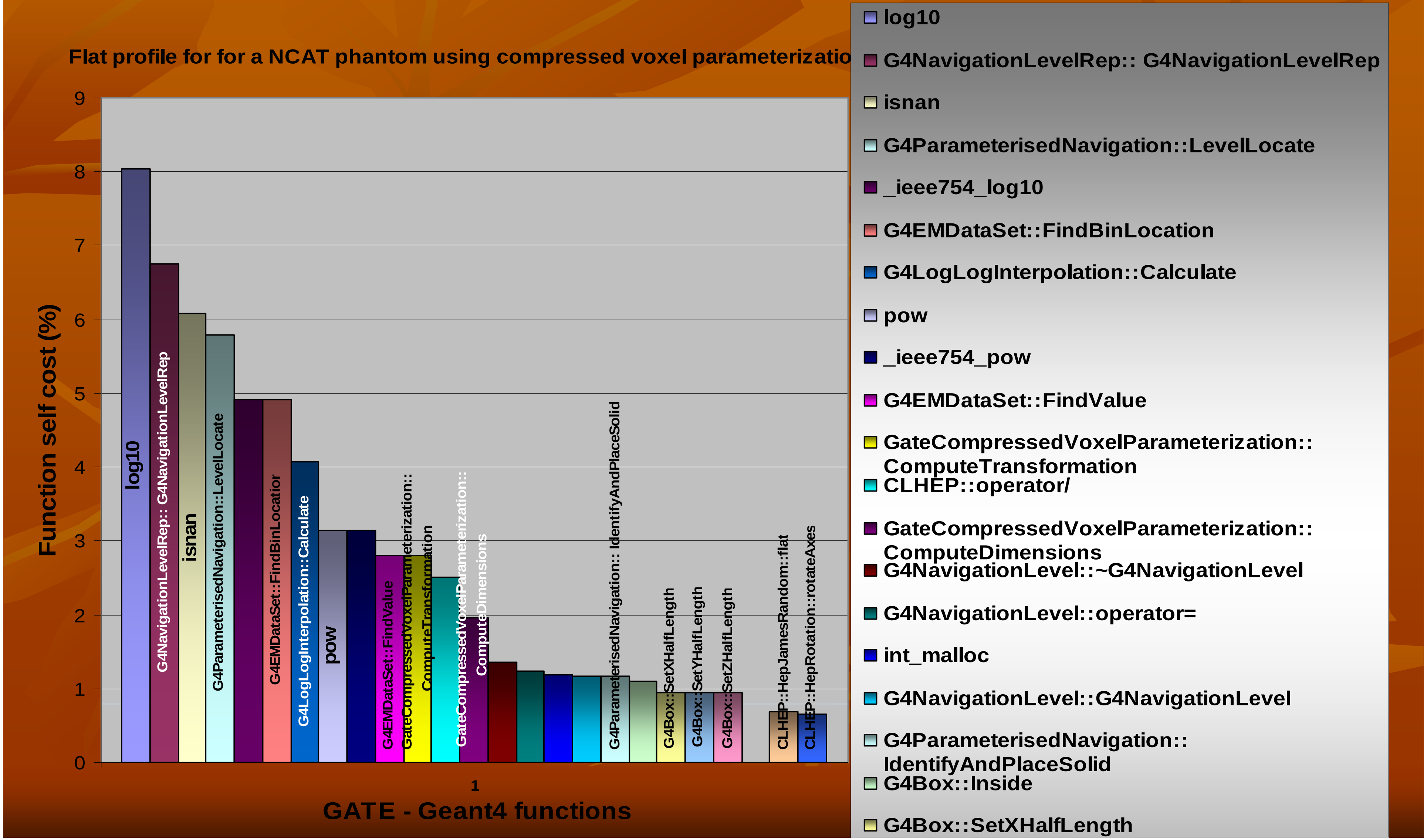
Callgrind study
analytical NEMA phantom



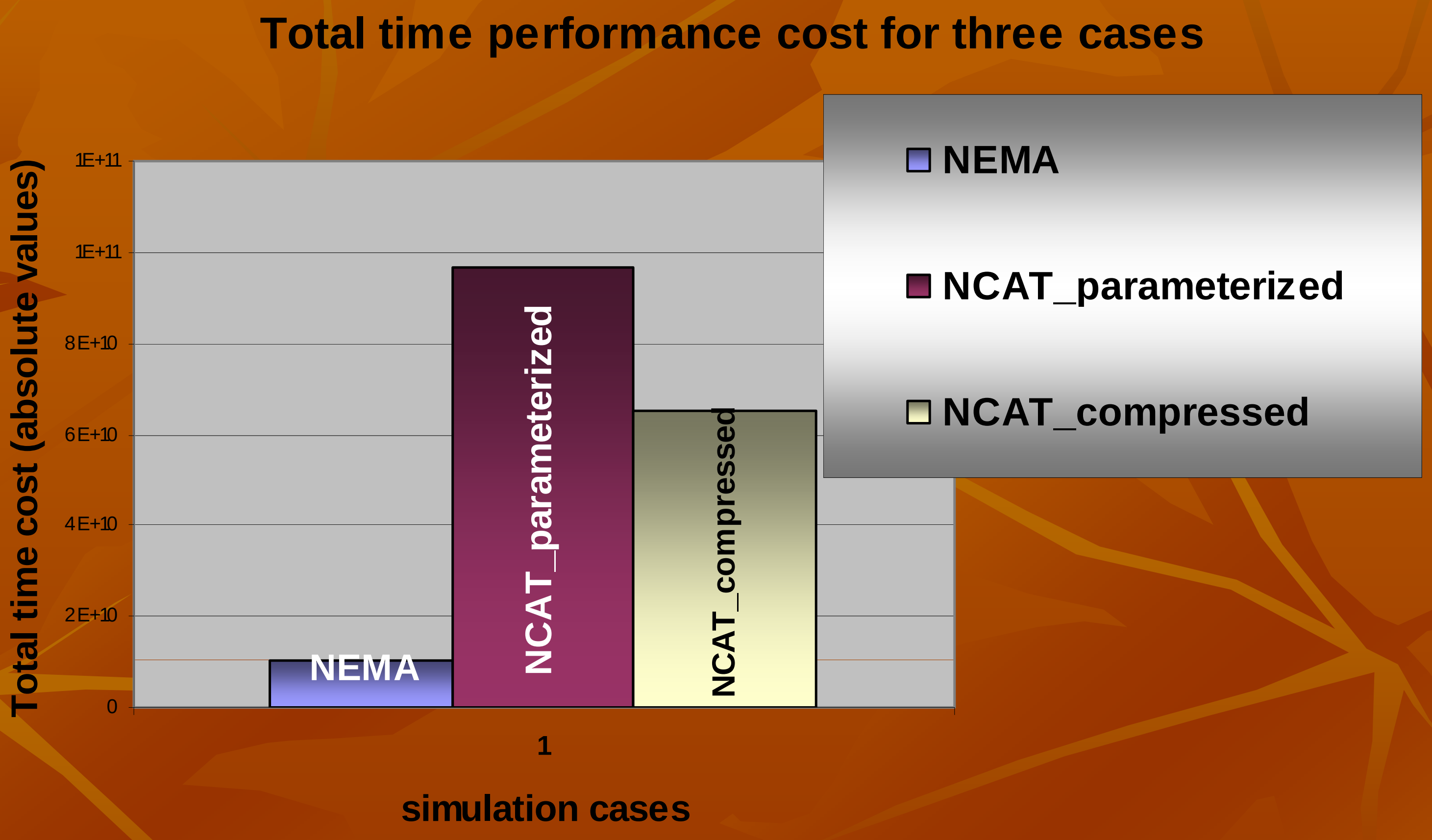
Callgrind study voxelized NCAT phantom (default parameterization)



Callgrind study voxelized NCAT phantom (compressed parameterization)



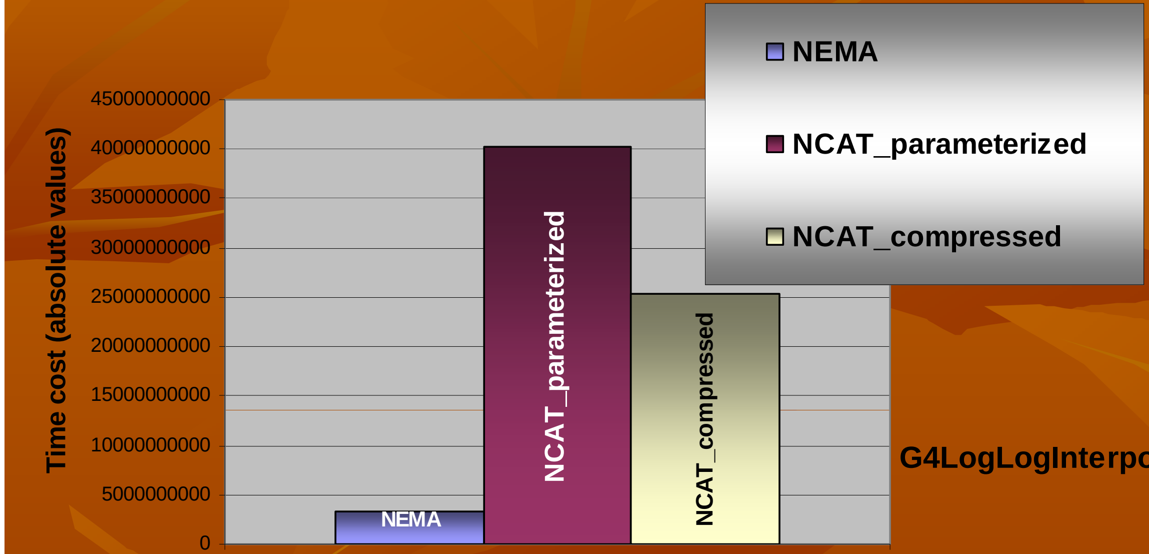
Callgrind study
Comparison of time-performance cost



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 time performance cost (inclusive) for three cases



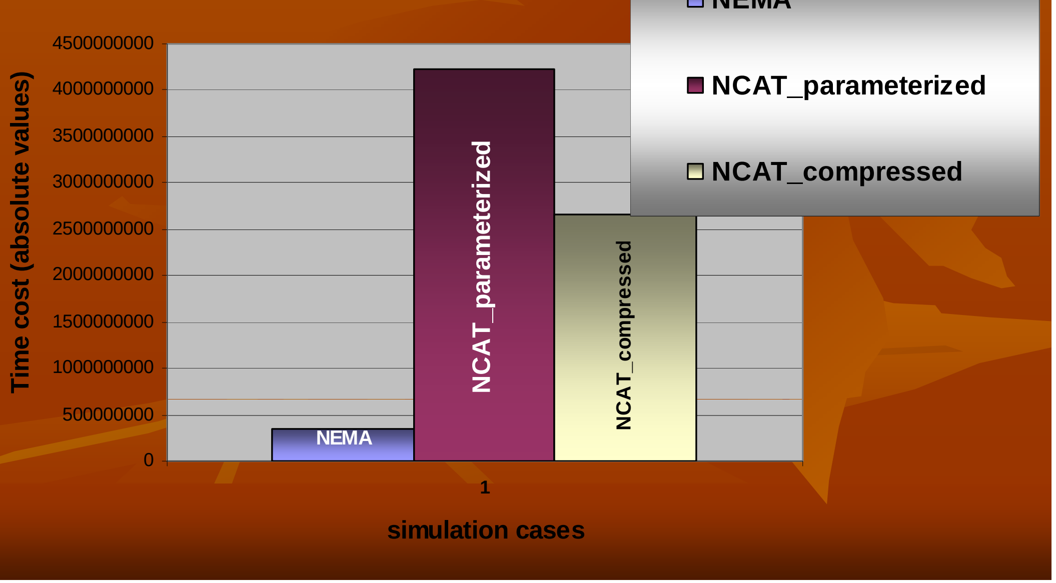
G4EMDataSet::FindValue exhibits one of the highest time inclusive costs (absolute values) among the G4functions which are called by the G4SteppingManager

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

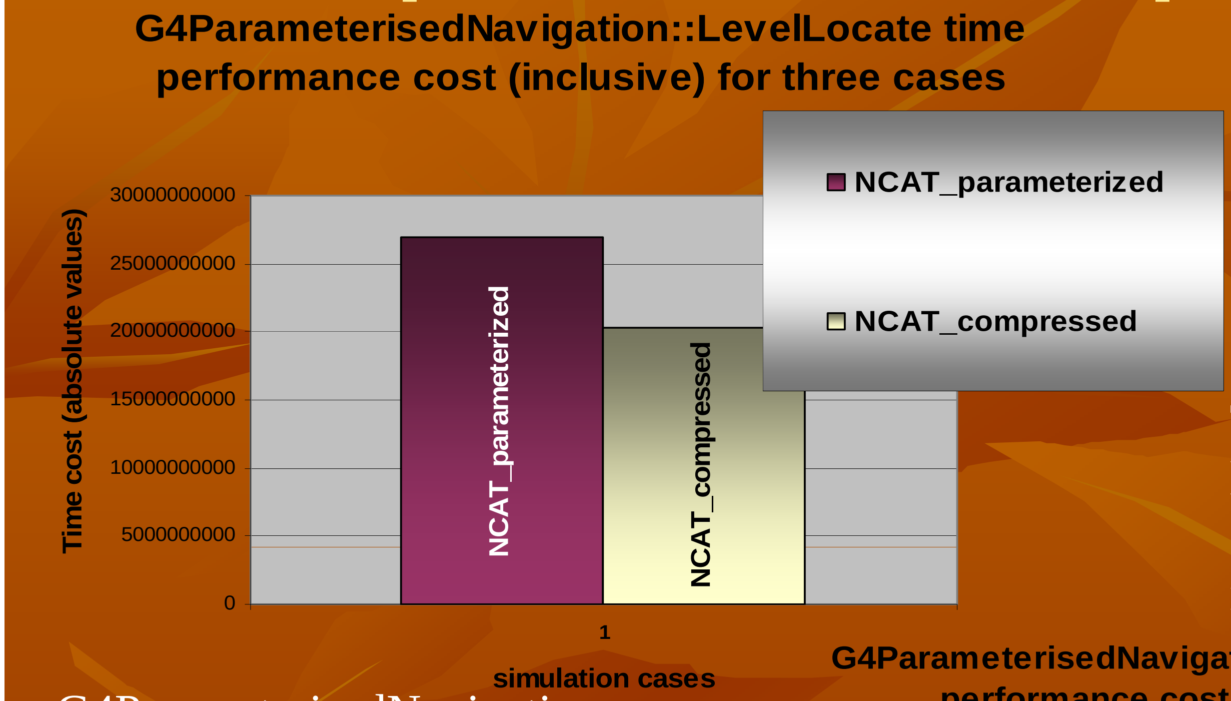
G4LogLogInterpolation::Calculate exhibits one of the highest time self costs (absolute values) among the G4functions which are called by the G4EMDataSet::FindValue

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

G4LogLogInterpolation::Calculate time performance cost (self) for three cases



Callgrind study
Comparison between default and compressed parameterization

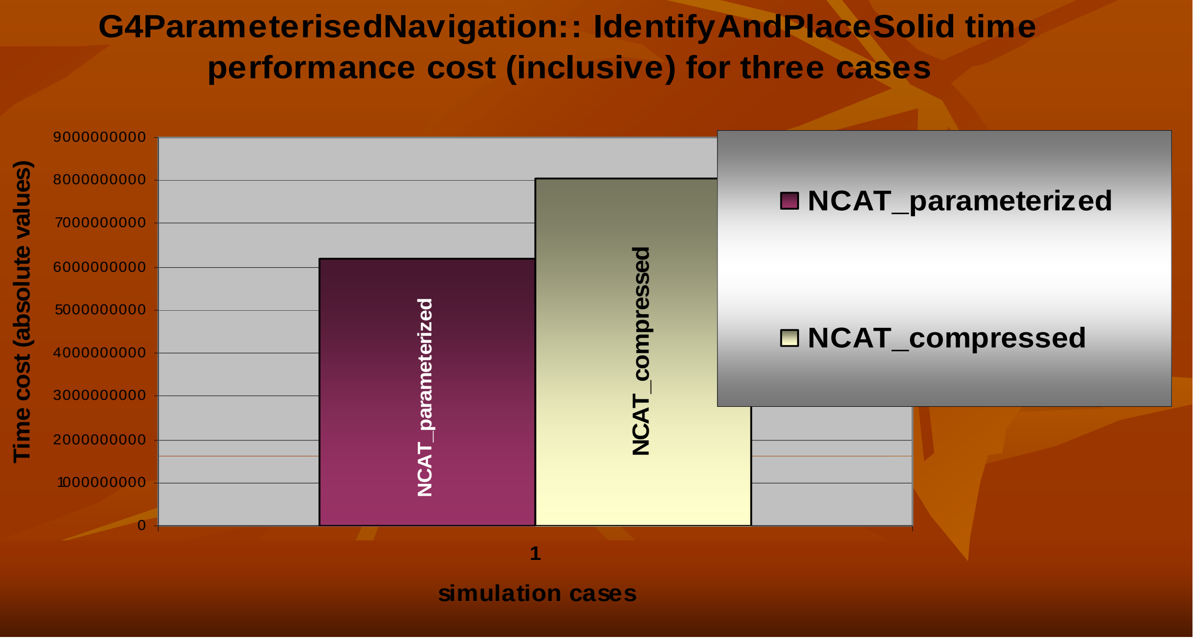


G4ParameterisedNavigation:: LevelLocate exhibits one of the highest time inclusive costs (absolute values) among the functions of the G4Navigator class

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

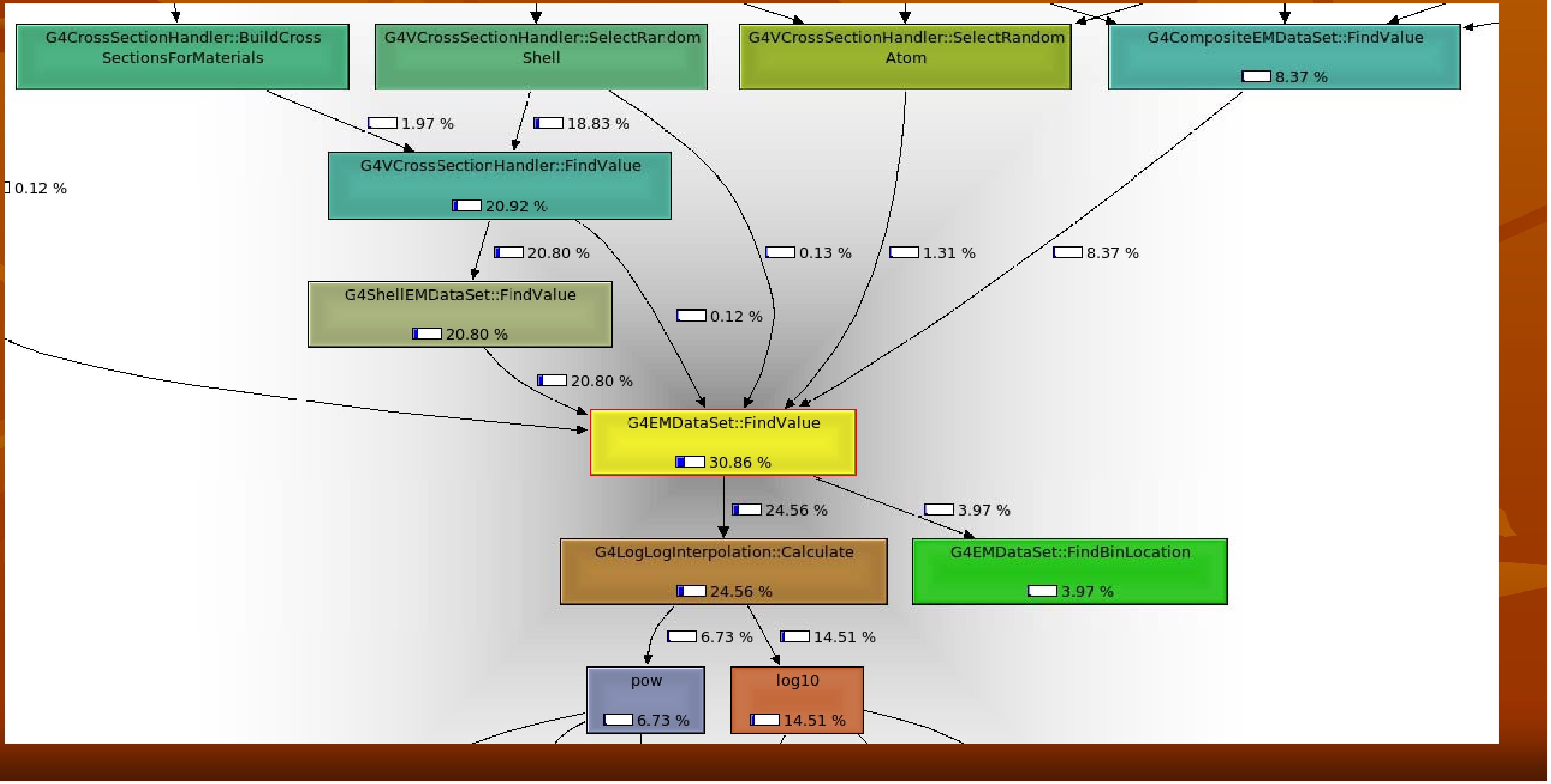
G4ParameterisedNavigation:: IdentifyAndPlaceSolid determines the parameterization type applied for each voxel

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



Callgrind study - Call graph NEMA analytical phantom

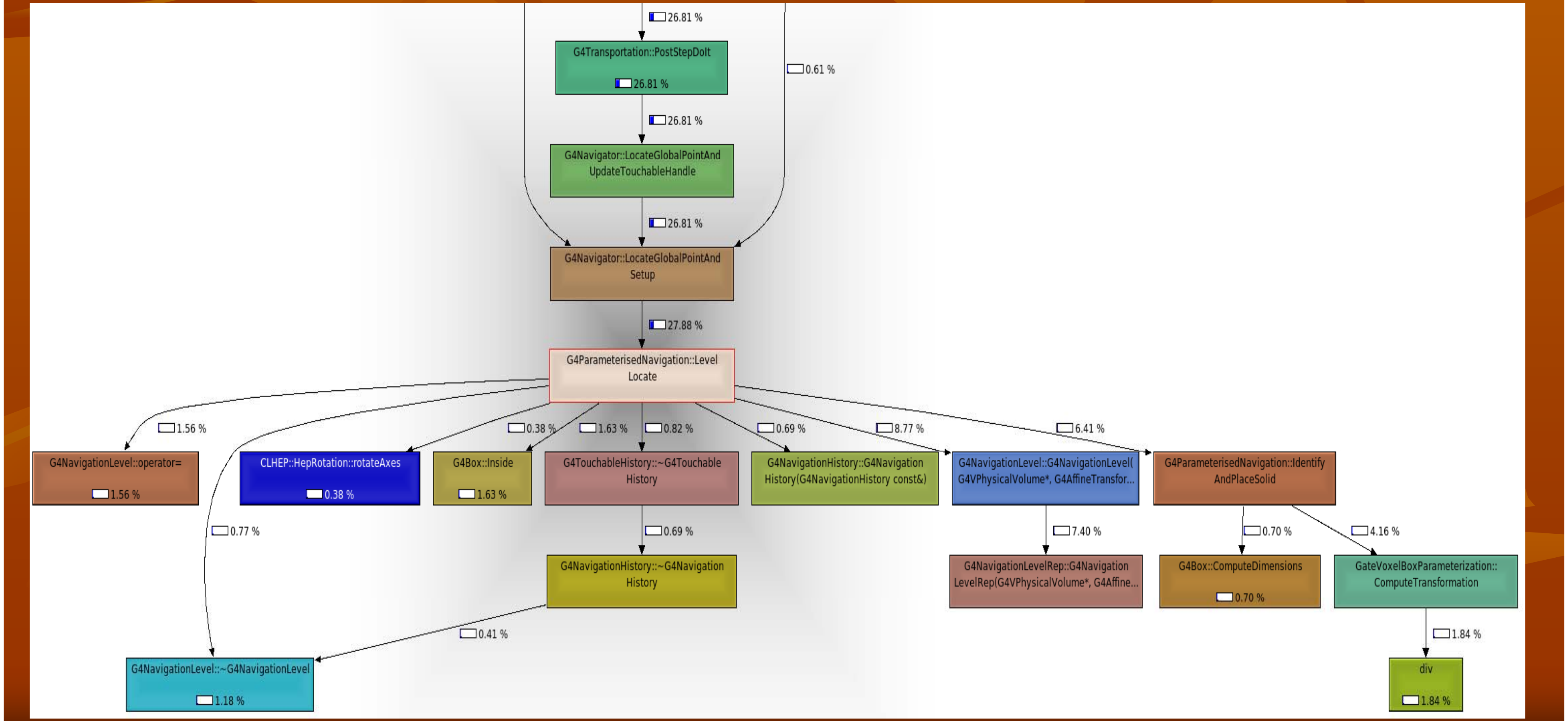
G4EMDataSet::FindValue function spends 30.86% of the total cost by completing it's calls to callees functions and by returning calls made to this function from other caller functions.



G4CrossSectionHandler::BuildCrossSectionsForMaterials
G4VCrossSectionHandler::SelectRandomShell
G4VCrossSectionHandler::SelectRandomAtom
G4CompositeEMDataSet::FindValue
8.37 %
1.97 %
18.83 %
G4VCrossSectionHandler::FindValue
20.92 %
0.12 %
20.80 %
0.13 %
1.31 %
8.37 %
G4ShellEMDataSet::FindValue
20.80 %
0.12 %
20.80 %
G4EMDataSet::FindValue
30.86 %
24.56 %
3.97 %
G4LogLogInterpolation::Calculate
24.56 %
G4EMDataSet::FindBinLocation
3.97 %
6.73 %
14.51 %
pow
6.73 %
log10
14.51 %

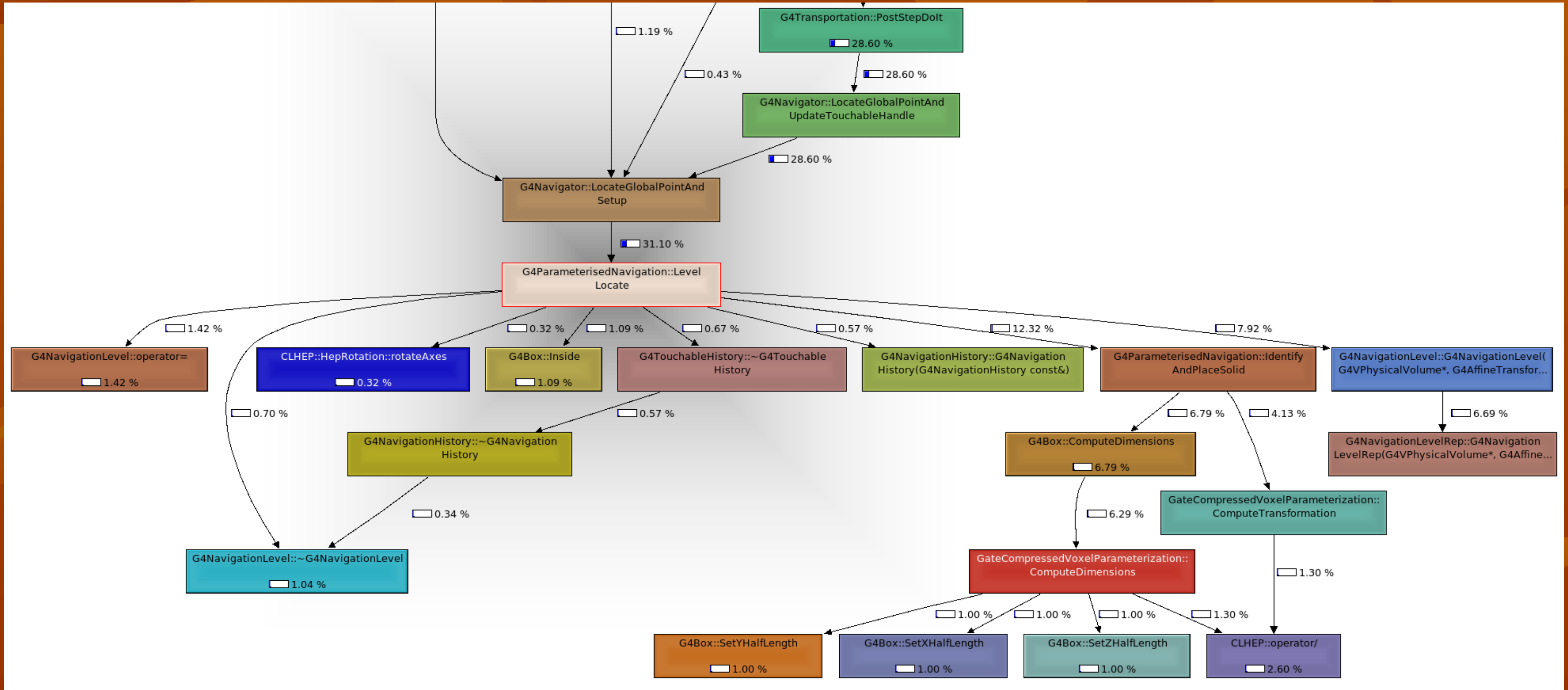
Callgrind study - Call graph
NCAT voxelized phantom (default parameterization)

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.


26.81 %
G4Transportation::PostStepDoIt
26.81 %
0.61 %
26.81 %
G4Navigator::LocateGlobalPointAndUpdateTouchableHandle
26.81 %
G4Navigator::LocateGlobalPointAndSetup
27.88 %
G4ParameterisedNavigation::LevelLocate
1.56 %
0.38 %
1.63 %
0.82 %
0.69 %
8.77 %
6.41 %
G4NavigationLevel::operator=
1.56 %
CLHEP::HepRotation::rotateAxes
0.38 %
G4Box::Inside
1.63 %
G4TouchableHistory::~G4TouchableHistory
G4NavigationHistory::G4NavigationHistory(G4NavigationHistory const&)
G4NavigationLevel::G4NavigationLevel(G4VPhysicalVolume*, G4AffineTransfor...
G4ParameterisedNavigation::IdentifyAndPlaceSolid
0.77 %
0.69 %
7.40 %
0.70 %
4.16 %
G4NavigationHistory::~G4NavigationHistory
G4NavigationLevelRep::G4NavigationLevelRep(G4VPhysicalVolume*, G4Affine...
G4Box::ComputeDimensions
0.70 %
GateVoxelBoxParameterization::ComputeTransformation
0.41 %
1.84 %
G4NavigationLevel::~G4NavigationLevel
1.18 %
div
1.84 %

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.



G4Transportation::PostStepDoIt
28.60 %
1.19 %
0.43 %
28.60 %
G4Navigator::LocateGlobalPointAndUpdateTouchableHandle
28.60 %
G4Navigator::LocateGlobalPointAndSetup
31.10 %
G4ParameterisedNavigation::LevelLocate
1.42 %
0.32 %
1.09 %
0.67 %
0.57 %
12.32 %
7.92 %
G4NavigationLevel::operator=
1.42 %
CLHEP::HepRotation::rotateAxes
0.32 %
G4Box::Inside
1.09 %
G4TouchableHistory::~G4TouchableHistory
G4NavigationHistory::G4NavigationHistory(G4NavigationHistory const&)
G4ParameterisedNavigation::IdentifyAndPlaceSolid
G4NavigationLevel::G4NavigationLevel(G4VPhysicalVolume*, G4AffineTransfor...
0.70 %
0.57 %
6.79 %
4.13 %
6.69 %
G4NavigationHistory::~G4NavigationHistory
G4Box::ComputeDimensions
6.79 %
G4NavigationLevelRep::G4NavigationLevelRep(G4VPhysicalVolume*, G4Affine...
0.34 %
6.29 %
GateCompressedVoxelParameterization::ComputeTransformation
G4NavigationLevel::~G4NavigationLevel
1.04 %
GateCompressedVoxelParameterization::ComputeDimensions
1.30 %
1.00 %
1.00 %
1.00 %
1.30 %
G4Box::SetYHalfLength
1.00 %
G4Box::SetXHalfLength
1.00 %
G4Box::SetZHalfLength
1.00 %
CLHEP::operator/
2.60 %

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%
  - 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
      - self: 3.24%, inclusive: 29.4% (for analytical phantoms)
      - self: 4.37%, inclusive: 33.18% (for analytical phantoms)

Current VRT Implementations (?)

- Variance Reduction Techniques (VRTs)
  - Importance sampling
    - Photon splitting
    - Russian roulette
  - Weight window sampling
  - Weight roulette
  - Scoring

- A number of VRT implementations are already available in Geant4, but
  - Some of the Geant4 classes can be optimized more,
  - Further classes implementing VRTs could be developed

Discussion of possible new VRTs applied

- 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
  - Estimation of speed-up factor
    - 3 – 4 times increase in efficiency of the application
- 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

- Collaboration between GATE and Geant4 VRTs workgroup
- Proposed targets of collaboration
  - Determination of the existing G4 classes need to be optimized
  - Determination of further G4 classes possibly needed to be implemented
  - Implementation of GATE-specific classes within GATE
  - Validation of the new classes themselves and within GATE-Geant4 frame
  - Publication of a GATE-specific patch for Geant4

Acknowledgements

- This work is a part of the project: fastGATE
  - funded by the Greek Secretary of Research and Technology
- Partners
  - National Technical University of Athens
  - University of California, Los Angeles