Geant4 based simulation of radiotherapy in CUDA

Koichi Murakami (KEK / CRC)

Stanford ICME, SLAC, G4-Japan Collaboration
supported by NVIDIA

The Collaboration

Geant4 @ SLAC

ICME
INSTITUTE for COMPUTATIONAL & MATHEMATICAL ENGINEERING at STANFORD UNIVERSITY

Geant4 @

NVIDIA CUDA

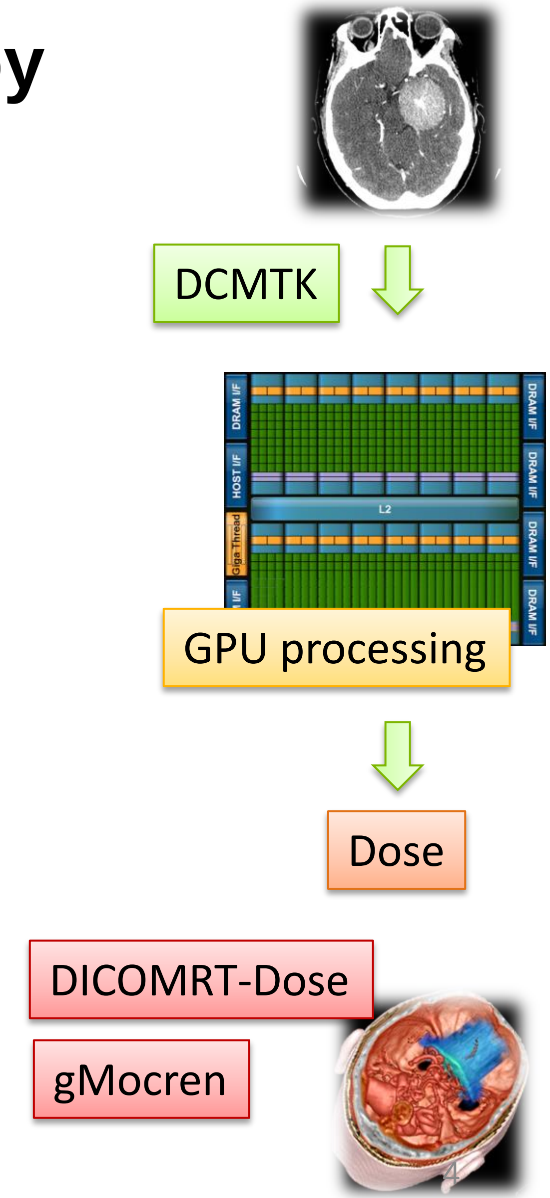
Special thanks to the CUDA Center of Excellence Program

Contents

G4CU Project Overview

Dose calculation for radiation therapy

- GPU-powered
  - parallel processing with *CUDA*
  - boost-up calculation speed
  - CUDA porting of Geant4
- Functions
  - voxel geometry
    - including DICOM interface
    - material : water with variable densities
  - limited Geant4 EM physic processes
    - electron/positron/gamma
  - scoring dose in each voxel



Physics Processes

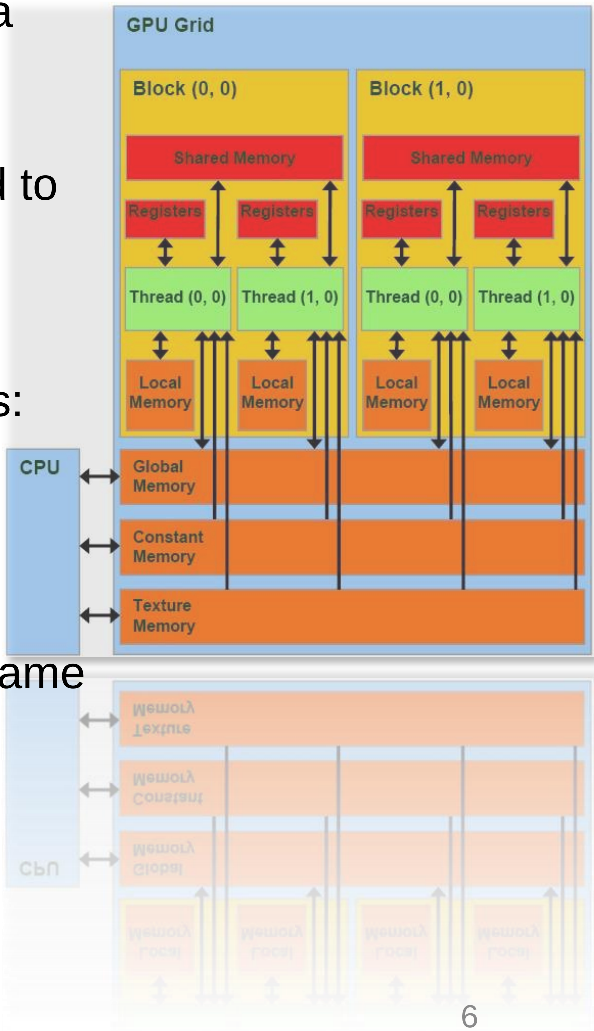
- particles : electron, positron, gamma
- energy range up to 100 MeV
- material: water with variable densities
- processes:

  electron / positron
    - energy loss (ionization, bremsstrahlung)
    - multiple scattering
    - positron annihilation

  gamma
    - Compton scattering
    - photo electric effect
    - gamma conversion
- physics tables
  - dE/dx, range, etc are retrieved from Geant4
  - Physics tables are prepared for "standard" water and rescaled with the density of each voxel.

CUDA Basics

- "**SIMD**" architecture : *Single Instruction, Multiple Data*
  - CUDA is a data parallel language
  - wants to run same instruction on multiple pieces of data
- Coalesced memory access
  - to maximize memory throughput, we want a single read to satisfy as many threads as possible
- Memory hierarchy
  - CUDA provides access to several device memory types:
    - global, shared, constant, texture
  - better memory usage for better performance
- Race conditions
  - arise when multiple CUDA threads attempt to write to same location in global memory
  - may happen in dose accumulation
  - we avoid race conditions or using atomic operations



Parallelization Challenges in Geant4

- Programming Methodology
  - CUDA porting of large and complex code
    - large scale of object-oriented design
    - inter-dependencies in many places
  - Branching, look-up tables, single-thread optimizations

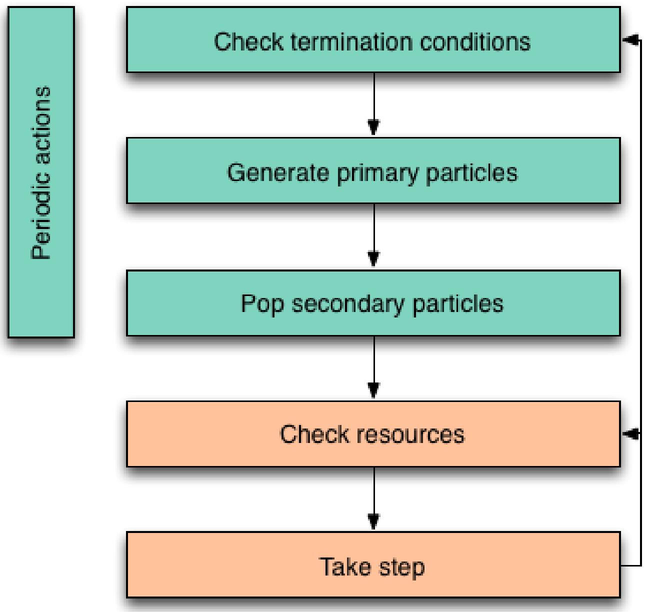
- Software Complexity
  - Sophisticated geometry and tracking management
  - Elaborate physics models

G4CU Basics

- Each GPU thread processes a single track until it dies or exits
  - GPU runs on 32k CUDA threads (256 threads on128bloks)
  - A track stores data for
    - particle spices, position, direction, energy, etc

- Each thread has two stacks
  - one for storing secondary particles
  - one for recording the energy dose in a voxel

- Periodic actions:
  - check termination conditions
  - generate primaries
  - pop secondary particles



Notes on Dose Accumulation

- 2 critical issues on performance:

- *Race conditions* might arise when multiple CUDA threads attempt to write to same location in global memory.
  - That may happen in dose accumulation in each voxel.
  - Two ideas were tried:
    - parallel stack for dose and reduction
    - `atomicAdd` operation in CUDA -> better performance
      - *explicit memory access*

- *Double precision* variables are used for dose.
  - other variables are single precision.
  - prevent from overflow
    - *small energy dep. + large accumulated dose*

Performance Profiling

nvpof is helpful to identify performance bottlenecks.

Process	Process total	Post step	PIL	Along step	Manag.	Init.	Step length	At-rest
Component total (%)	**100.00**	**52.80**	**20.73**	**14.16**	**4.19**	**3.78**	**3.46**	**0.89**
Bremsstrahlung	**32.81**	29.83	2.23			0.75		
Ionization	**14.83**	1.70	3.50	8.84		0.79		
Photo-electric effect	**10.79**	8.80	1.57			0.41		
Gamma conversion	**10.67**	8.72	1.54			0.41		
Multiple scattering	**10.50**		3.43	4.58			2.49	
Transport	**8.67**	1.17	5.23	0.74		0.57	0.96	
Compton scattering	**4.20**	2.14	1.58			0.48		
Management	**4.19**				4.19			
Pair production	**2.56**	0.44	1.23			0.36		0.53
Electron deletion	**0.79**		0.43					0.36

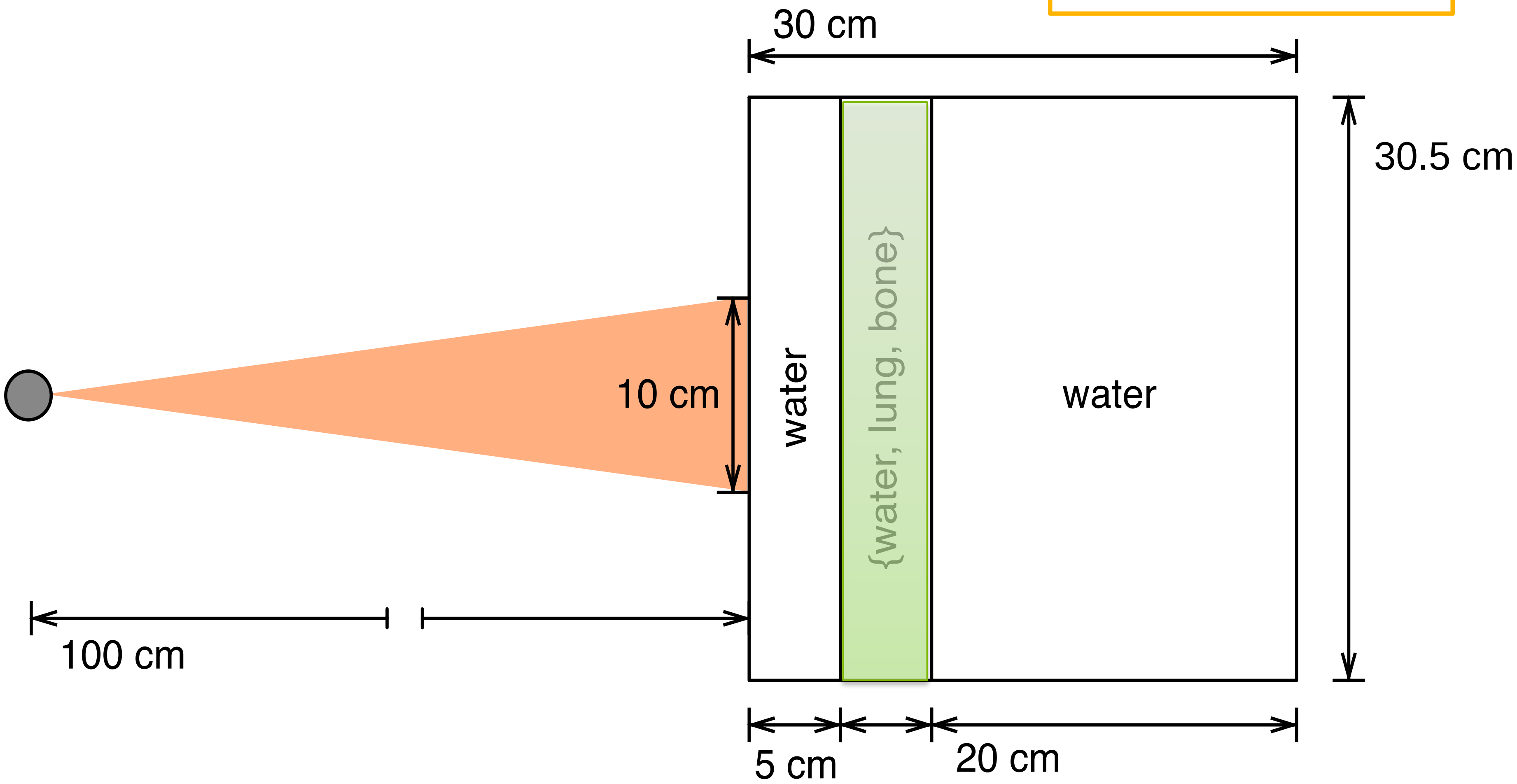
Hardware & SDK

- GPU:
  - Tesla K20 (Kepler)
  - 2496 cores, 706 MHz, 5GB GDDR5 (ECC)

- SDK:
  - CUDA 5.5, CURAND
- CPU:
  - Xeon X5680 (3.33GHz)

Phantom Geometry Configurations

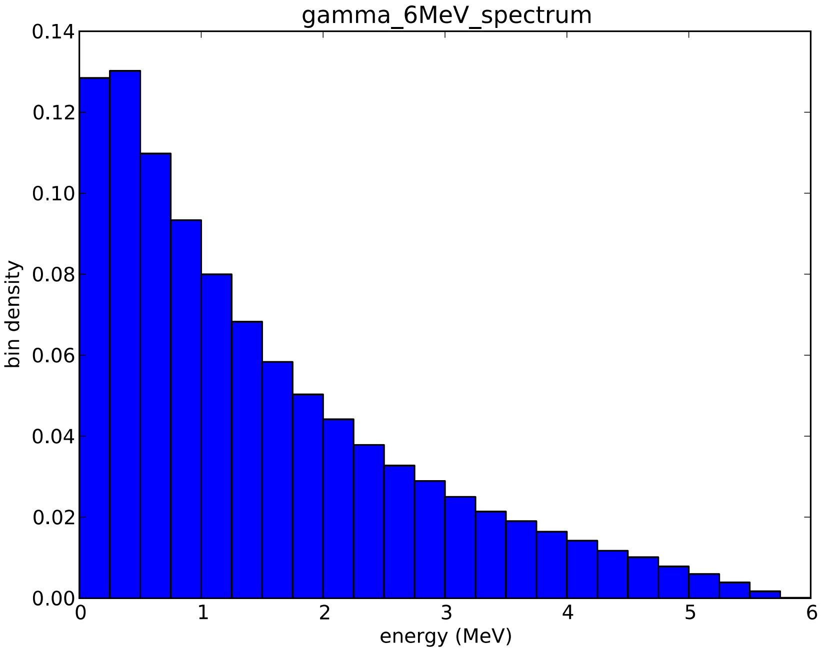
voxel size: 5x5x2mm

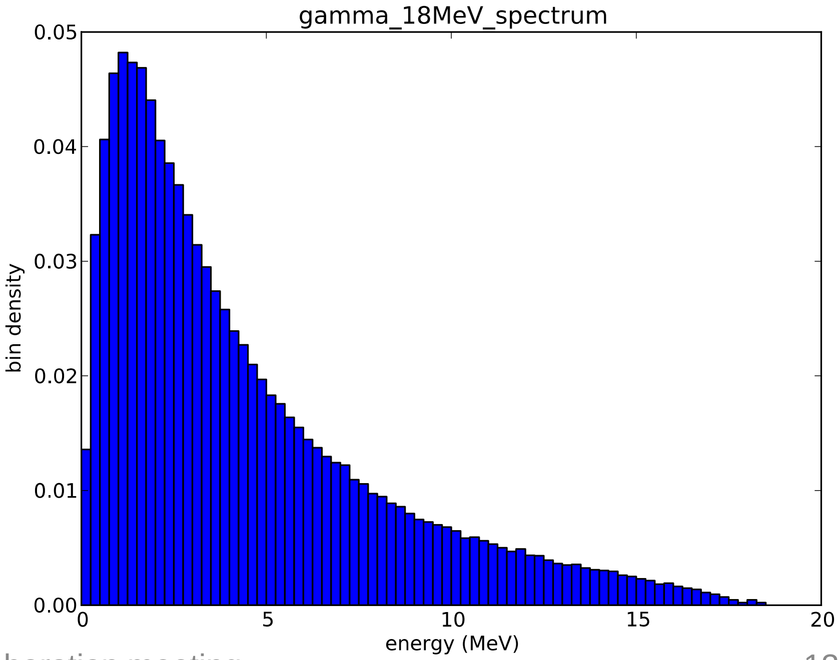


Particle Sources

- Mono-energetic source :
  - 20 MeV electron
  - 6 MeV photons

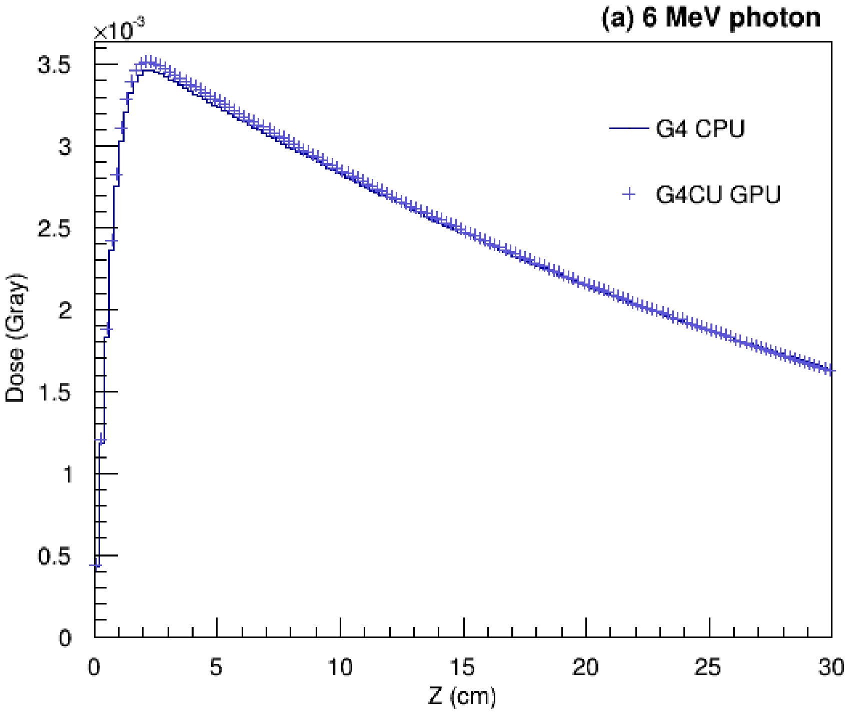
- Spread energy source generated by medical linac
  - 6 MV & 18 MV photons

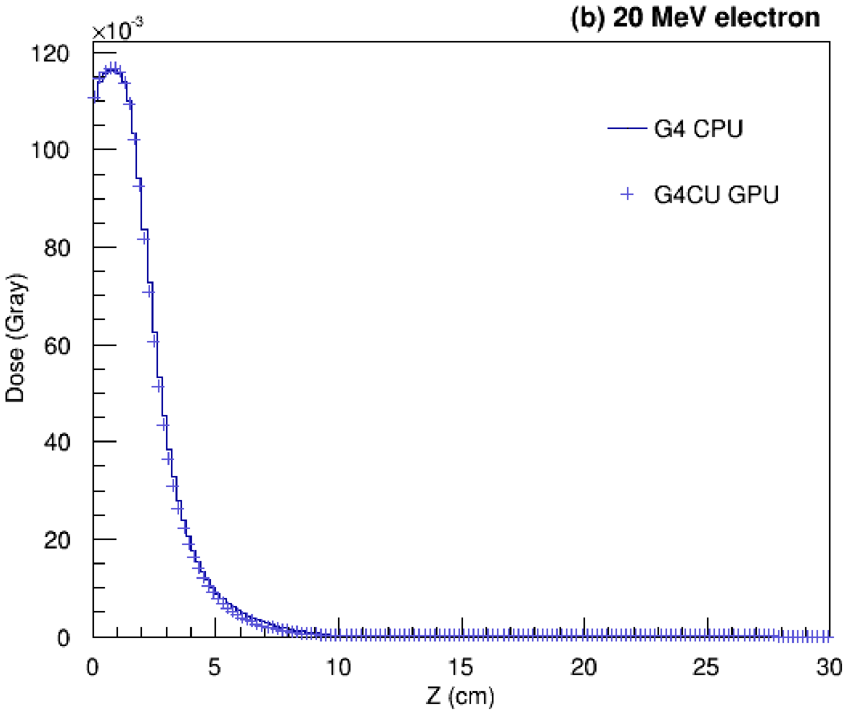




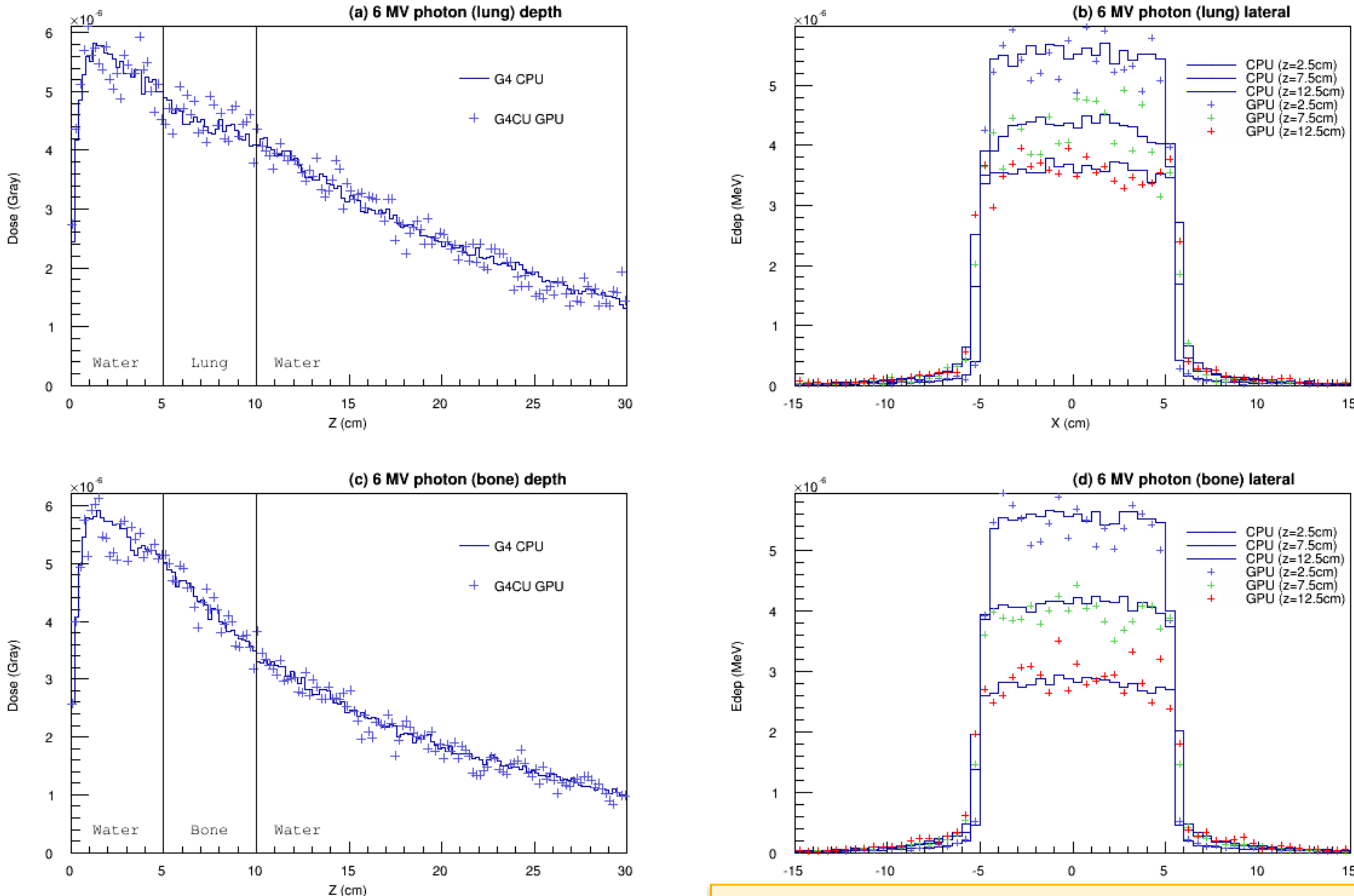
Depth Dose Distribution, CPU vs. GPU

- Depth dose distribution for water phantom
  - 6MeV photon and 20 MeV electron (pencil beam)
  - depth dose along central voxels
  - 100M primaries





Comparisons for Slab Phantoms



- Lung/Bone as an inner material
- 6MV photon
- 100M(CPU) vs 1M(GPU)

Computation Time of Geant4/CPU and G4CU/GPU

Primary	Phantom	Time/History CPU (sec)	Time/History GPU (sec)	CPU/GPU
20 MeV electron	Water	1.06E-03	2.52E-05	*42.1*
6 MeV photon	Water	4.47E-04	1.12E-05	*39.9*

CPU: Intel Xeon X5680 (3.33 GHz)
GPU : NVIDIA Tesla K20

G4CU achieves about *40 times speedup* against CPU-based Geant4 simulation.

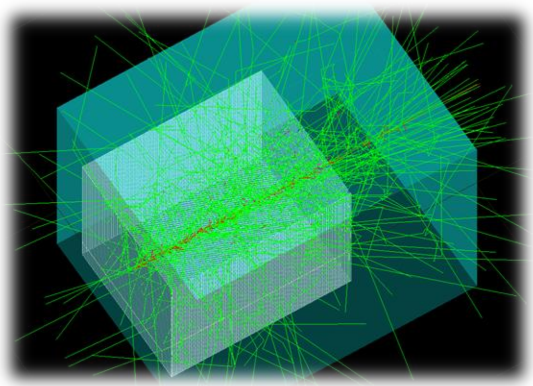
Software workflow

water phantom

DICOM Data | DCMTK

G4CU voxel geom. | dose output

G4 voxel geom.



Visualization

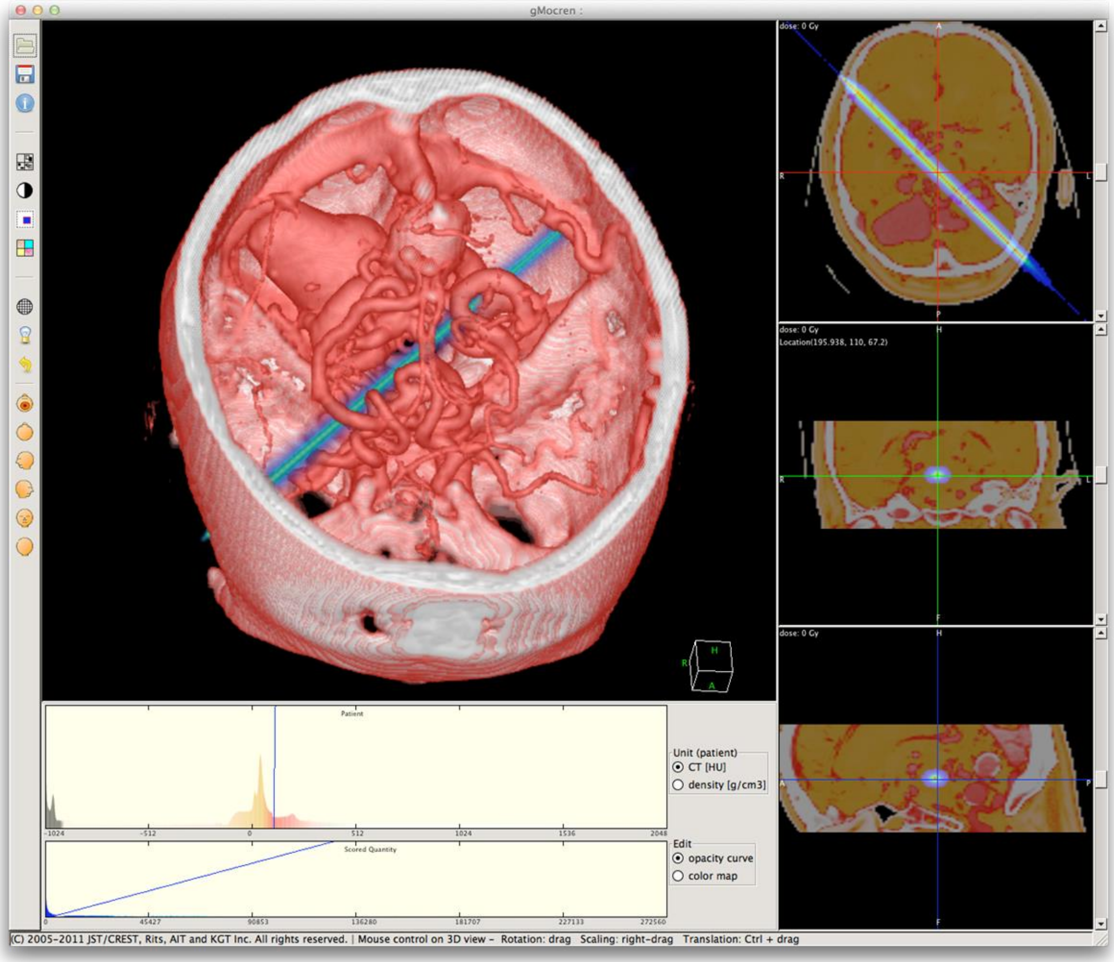
GDD IO | GDD file

ROOT | ROOT file

Histogramming

Visualization with gMocren

50M 6MeV photons, pencil beam shot on head



voxel size: 1.7 x 1.7 x 1.2 mm
segments: 128 x 128 x 64
≈ 1M voxels

Just for Demonstration

Summary

- Collaborative activity on Geant4-GPU between
  - Stanford ICME, SLAC, and G4-Japan (KEK), supported by NVIDIA
- Focused on medical application
  - Dose calculation in voxel domain
  - Geant4 EM physics processes
- CUDA porting of Geant4
  - Core part of implementation was done.
  - Verification against CPU version of Geant4
    - well-agreed in 1st order
  - Performance gain about 40 times
- Several ongoing follow-ups
  - robustness, performance improvements, improved code