Optimization of CUDA-based Monte Carlo Simulation for Radiation Therapy

GTC 2014

N. Henderson & K. Murakami

The collaboration

Geant4 @ SLAC

Geant4 @

ICME

INSTITUTE for COMPUTATIONAL & MATHEMATICAL ENGINEERING at STANFORD UNIVERSITY

NVIDIA

CUDA

Special thanks to the CUDA Center of Excellence Program

Makoto Asai, SLAC
Joseph Perl, SLAC
Andrea Dotti, SLAC
Koichi Murakami, KEK
Takashi Sasaki, KEK
Margot Gerritsen, ICME
Nick Henderson, ICME

Radiotherapy simulation methods

Analytic

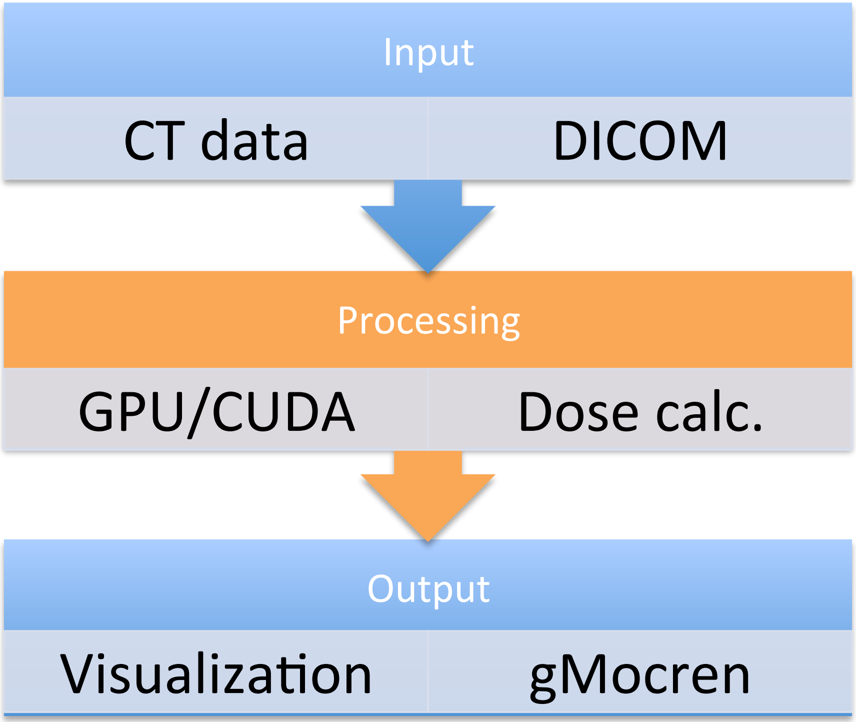
- time: seconds to minutes
- accurate within 3-5%
- used in treatment planning

Monte Carlo

- time: several hours to days of CPU time
- accurate within 1-2%
- used to verify treatment plans in certain cases

Project overview

GPU based dose calculation for radiation therapy



Input

CT data

DICOM

Processing

GPU/CUDA

Dose calc.

Output

Visualization

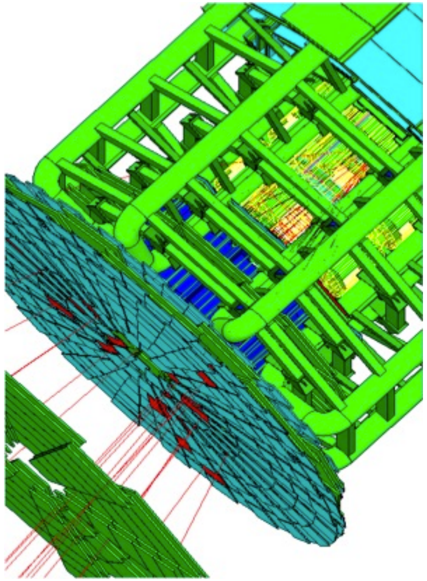
gMocren

Features

- GPU code based on Geant4
- Voxelized geometry
- DICOM interface
- Material is water with variable density
- Limited Geant4 EM physics: electron/positron/gamma
- Scoring of dose in each voxel
- Process secondary particles

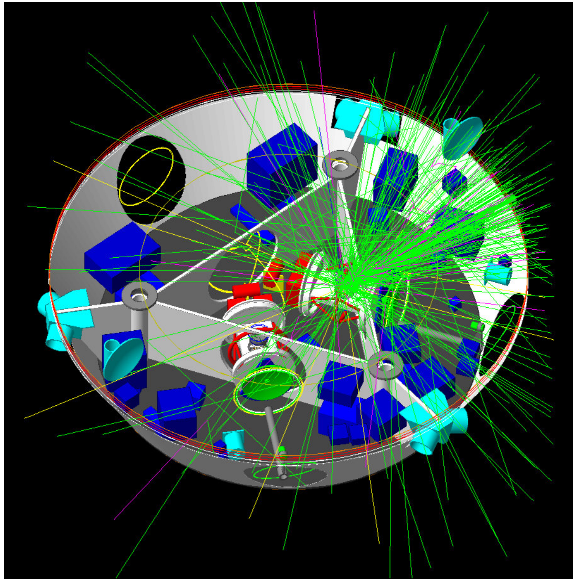
Geant4

High Energy Physics



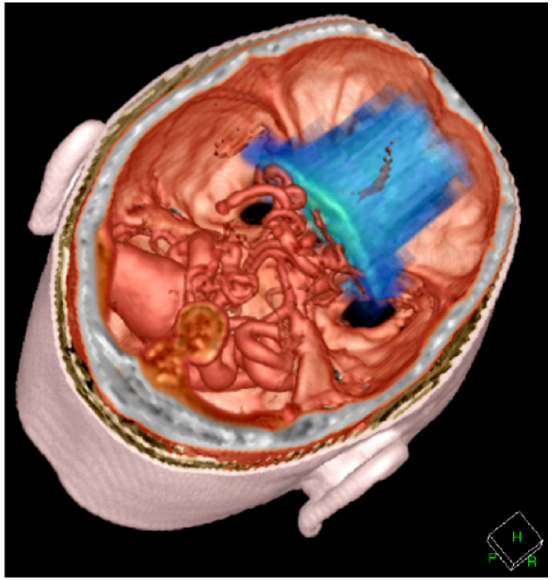
ATLAS

Space & Radiation



LISA

Medical Physics

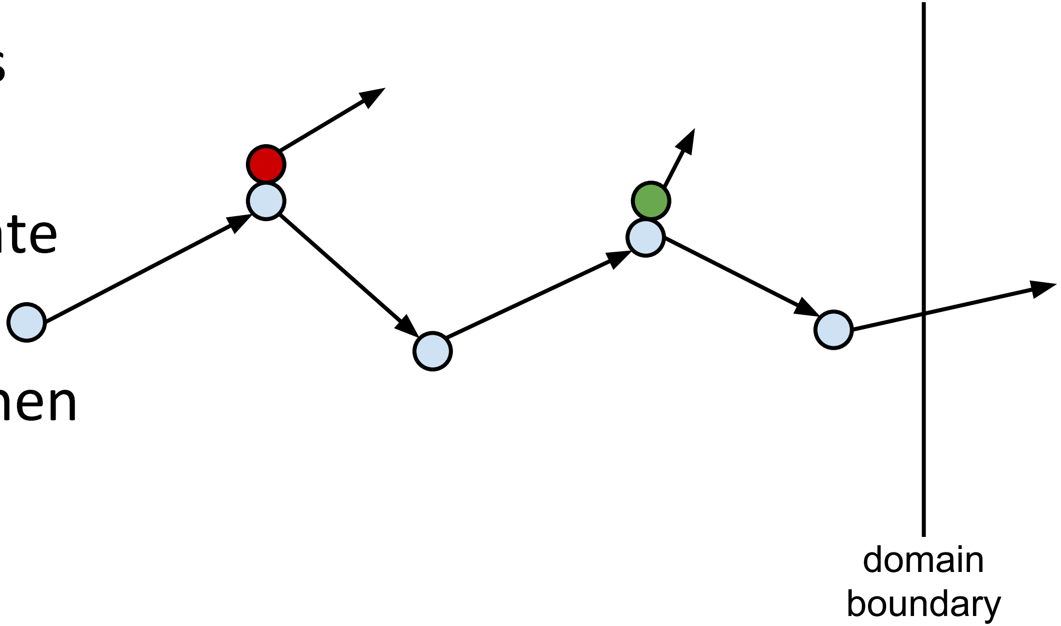


gMocren

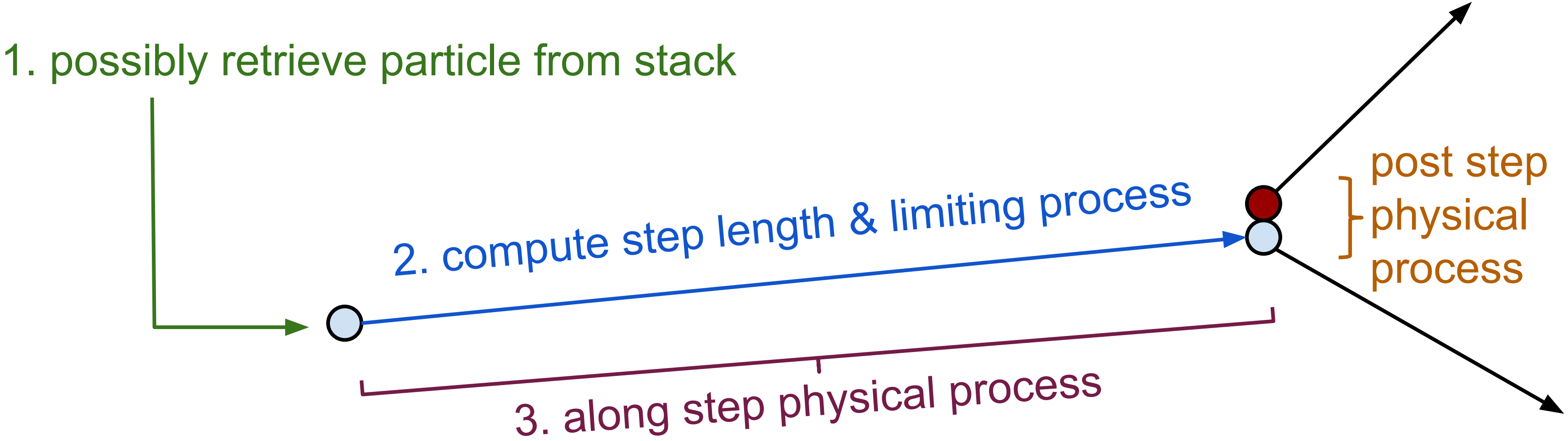
Images from: [Geant4 gallery]() and [gMocren]()

Basic idea

- Particles are transported through material in steps
- Physics processes act on particles and may generate secondaries
- Particles are removed when out of domain or out of energy

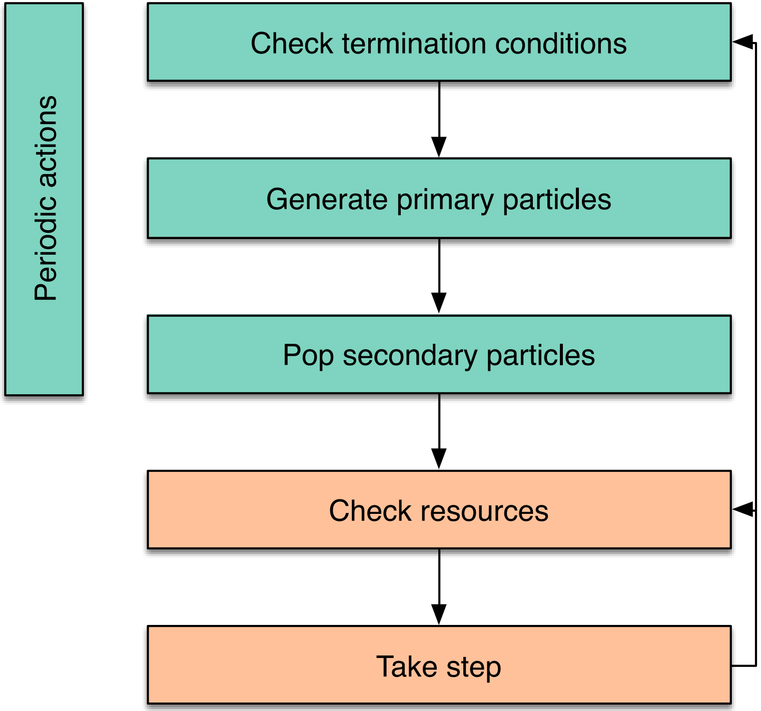


A single step

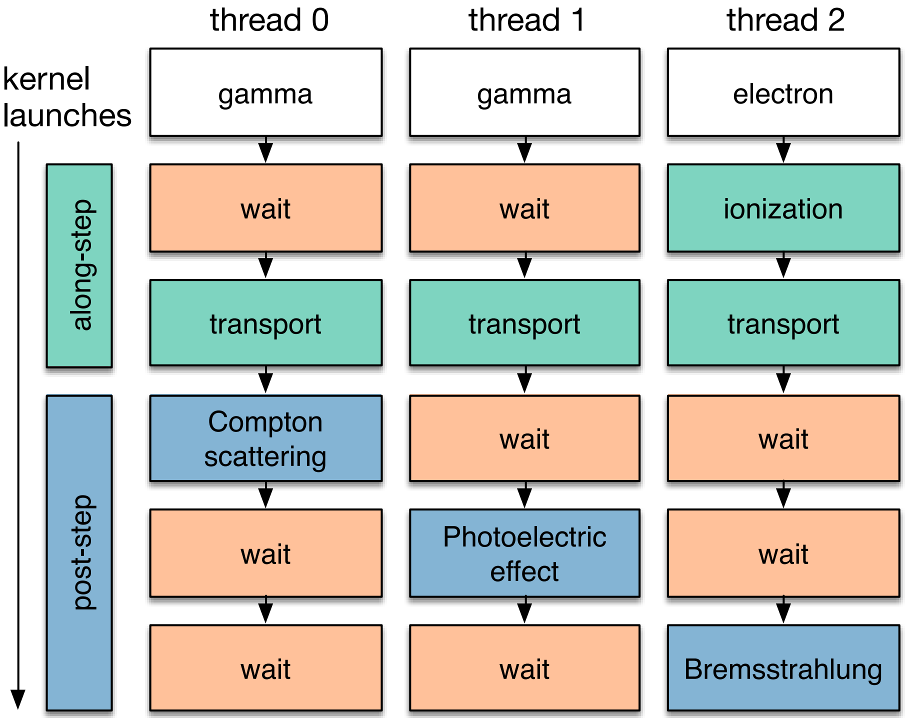

1. possibly retrieve particle from stack
2. compute step length & limiting process
post step physical process
3. along step physical process

G4CU: CUDA implementation

- Each GPU thread processes a single particle until the particle exits the geometry
- Each thread has a stack for secondary particles
- Every thread takes a step in each iteration
- Algorithm is periodically interrupted for various actions



G4CU: parallel execution



kernel launches	thread 0	thread 1	thread 2
	gamma	gamma	electron
along-step	wait	wait	ionization
	transport	transport	transport
post-step	Compton scattering	wait	wait
	wait	Photoelectric effect	wait
	wait	wait	Bremsstrahlung

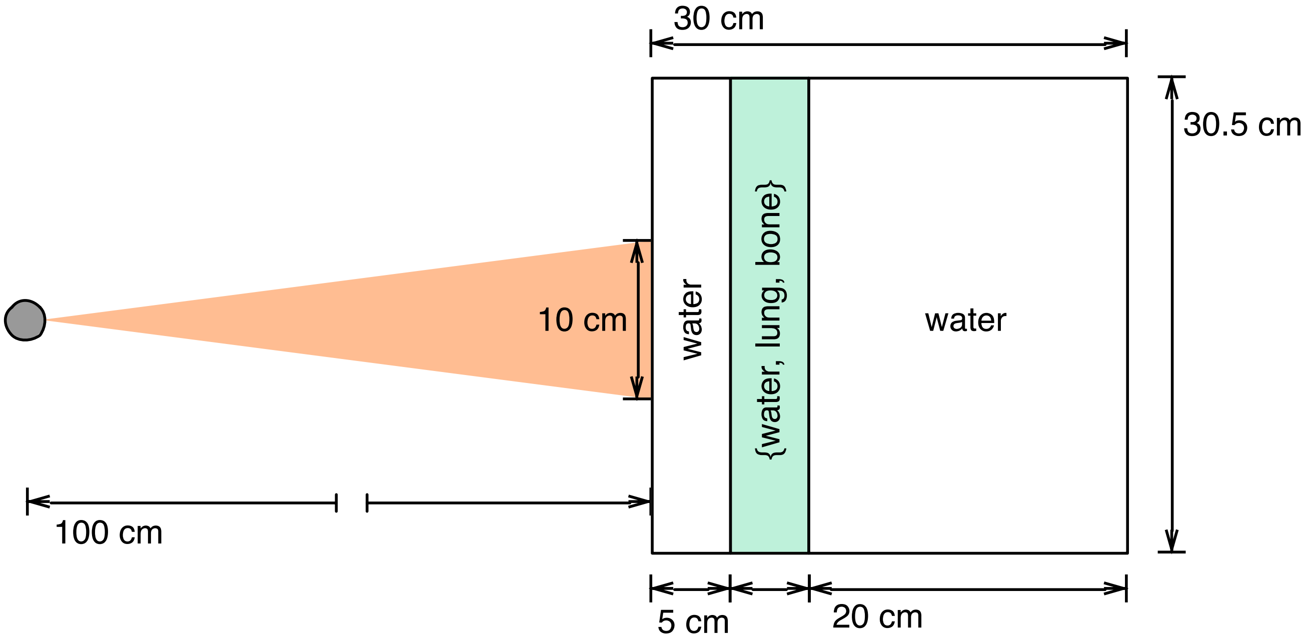
- CUDA kernels are applied to all particles
- Threads must wait if physics process does not apply to particle
- Parallel execution is achieved:
  - maintenance operations
  - transport process
  - same process on same particle

Performance and validation

Hardware and software:

- GPU:
  - Tesla K20 Kepler
  - 2496 cores, 706 MHz, 5GB GDDR5 (ECC)
- CPU
  - Xeon X5680 (3.33GHz)
  - single thread operation
- SDK:
  - CUDA 5.5
  - Geant4 release 9.6 p2

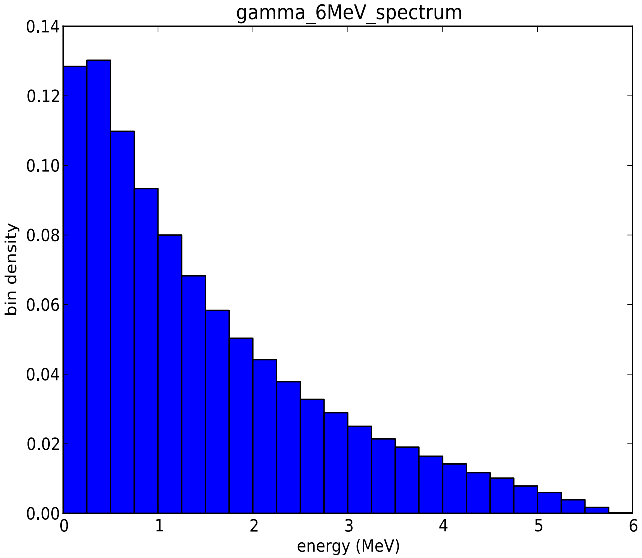
Phantom geometry

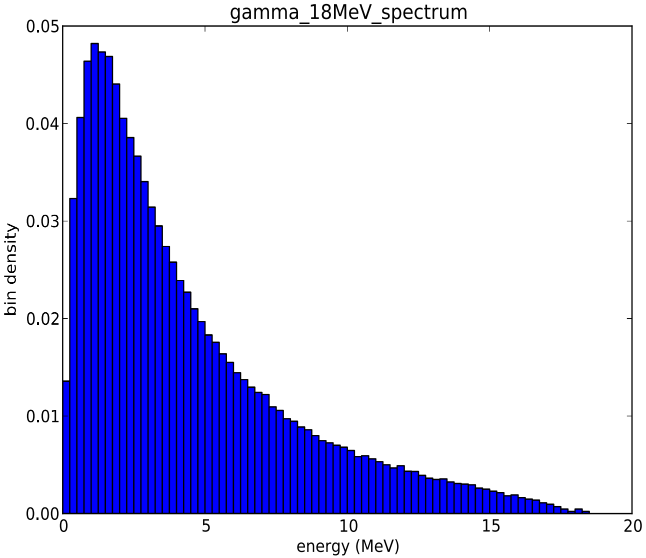


voxel size: 5mm x 5mm x 2mm

Particle sources

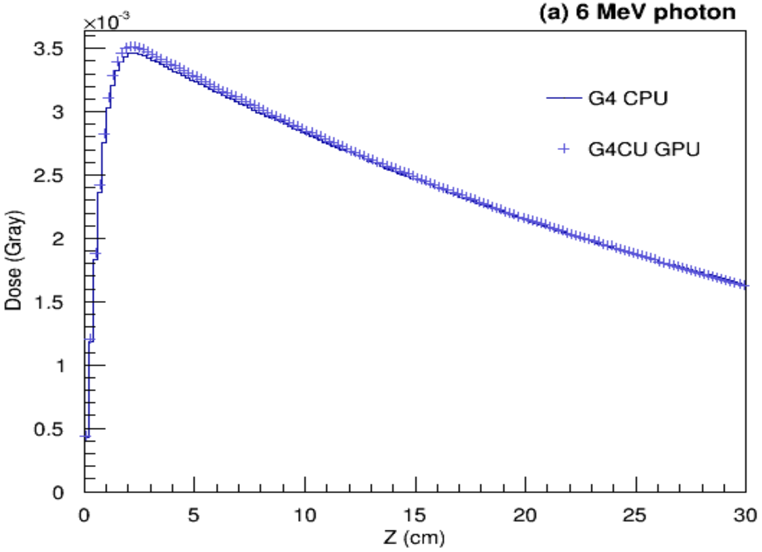
- 20 MeV electron / 6 MeV photons (mono-energy)
- 6 MV & 18 MV spectrum photons

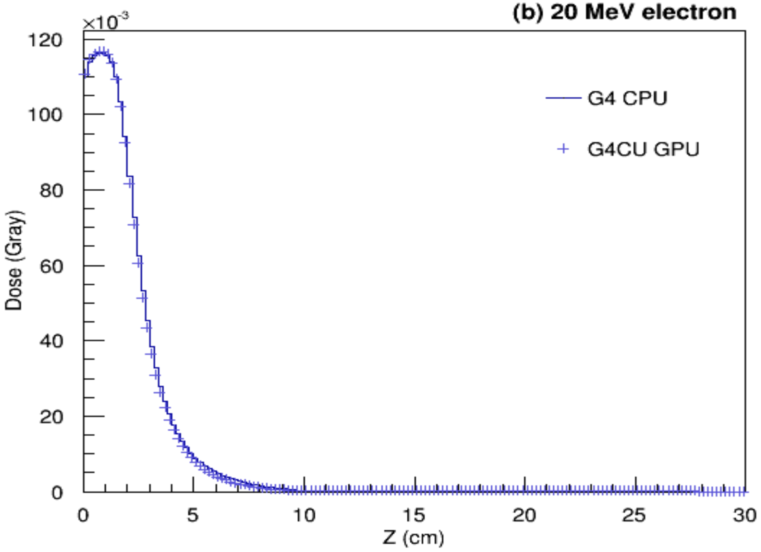




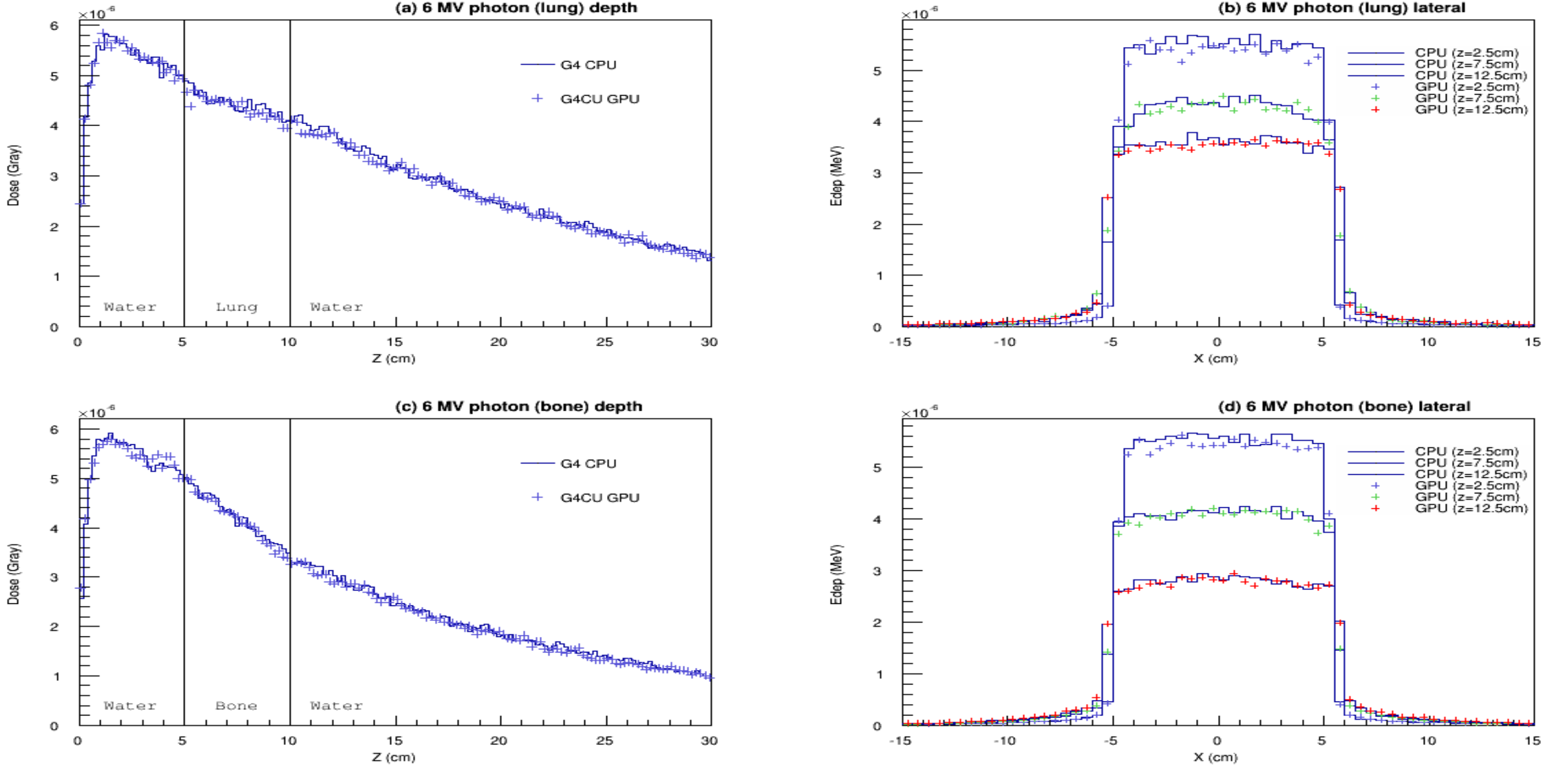
Depth dose distribution: water phantom

- 6MeV photon, 20 MeV electron
- dose along central axis



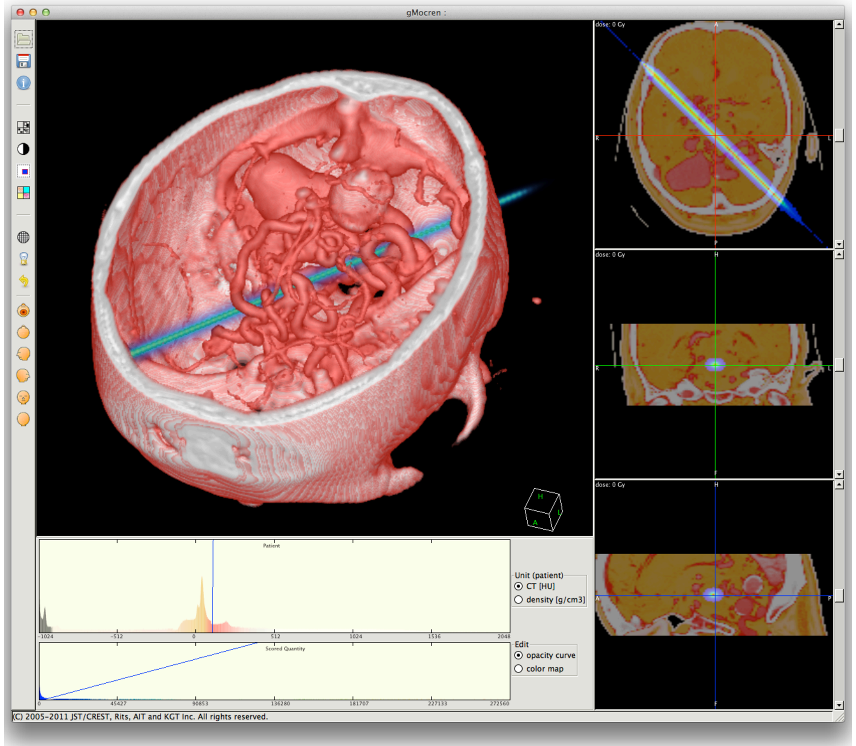


Comparisons for slab phantoms



6MV photon, Lung/Bone as inner slab material

Visualization with gMocren



Unit (patient)
● CT [HU]
○ density [g/cm3]

Edit
● opacity curve
○ color map

(C) 2005-2011 JST/CREST, Rits, AIT and KGT Inc. All rights reserved.

- Image for demonstration purposes only!
- 50M 6MeV photons
- Pencil beam configuration

Challenges

- Geant4 is complex
  - Implement targeted subset
- Varied character of physics process
  - Bad occupancy
  - No clear optimization target
- Random simulation
  - Thread divergence
- Lookup (interpolation) tables
  - Thread divergence
  - Non-ideal memory access
- Energy deposition in global array
  - Possible race condition
  - Non-ideal memory access

Physics Processes

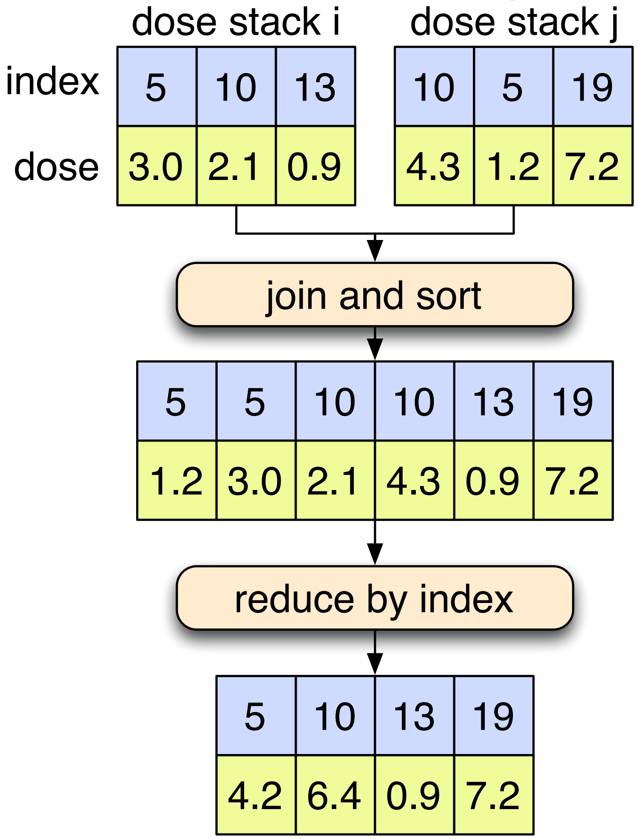
Process	Lines of code	Initialization kernel registers	Step length kernel registers	Action kernel registers
Compton scattering	116	14	18	49
Photo electric effect	118	14	18	40
Gamma conversion	160	14	18	36
Bremsstrahlung	156	14	18	44
Ionization	394	14	(18,18)	(60,36)
Scattering	575		12	28
Positron annihilation	156		12	26
Electron removal	44		12	8
Transport	491	20	22	(20,22)

Physics Profile (6 MeV gamma)

Sum of Time(%)	step	along-step	after-step	other	init	at-rest	Grand Total
em-helper	18.4				4.2		22.5
ionization	2.5	15.3	2.1				19.9
multiple-scattering	11.6	7.7					19.3
management	3.7			8.7	0.5		12.8
transport	2.4	1.7	3.3		1.4		8.9
bremsstrahlung			5.7				5.7
positron-annihilation	2.1		0.9		0.6	0.7	4.3
compton-scattering			2.6				2.6
gamma-conversion			1.4				1.4
photo-electric-effect			1.3				1.3
electron-removal						1.2	1.2
Grand Total	**40.7**	**24.6**	**17.4**	**8.7**	**6.6**	**2.0**	**100.0**

Optimization 1: dose storage

- Old idea: use thrust to join and reduce dose stacks into global array
- Better idea: use `atomicAdd`
- Speedup: ~1.5 (at the time)



dose stack i

index	5	10	13
dose	3.0	2.1	0.9

dose stack j

index	10	5	19
dose	4.3	1.2	7.2

join and sort

5	5	10	10	13	19
1.2	3.0	2.1	4.3	0.9	7.2

reduce by index

5	10	13	19
4.2	6.4	0.9	7.2

Optimization 2: device configuration

- 6 MeV gamma pencil beam
- 3,000,000 events
- Table shows run time / shortest time
- Voxels: 61 x 61 x 150
- 1 = 22 seconds
- ~ 136 events / ms

	threads per block				
blocks	32	64	128	256	512
32	17.98	9.85	5.62	3.21	2.04
64	9.85	5.63	3.2	2	1.48
128	5.62	3.21	1.98	1.39	1.22
256	3.55	2.15	1.36	1.1	1.14
512	2.42	1.46	1.08	1.02	1.15
1024	1.8	1.22	**1**	1.01	1.61

	threads per block		
blocks	32	64	128
1000	2.2	1.33	**1**
2000	1.82	1.19	1.03
3000	1.8	1.19	1.04
4000	1.74	1.27	1.15

Other optimizations

- New hardware!
- Refactoring
- New features sometime slow us down
- Results from:
  - 512 x 512 x 256 geometry
  - 6 MeV gamma pencil beam

Date	Hardware	Feature	Time (min)	Events / ms
March	C2070		72	23.1
May	K20		60	27.8
June	K20	atomicAdd	41	40.7
July	K20	multiple scattering	42.5	39.2
August	K20	world volume	49	34.0
Current	K20	Refactoring, device config	23	72.5

Comparison to Geant4 on CPU

- Geometry:61 x 61 x 150
- GPU: Tesla K20
- CPU: Xeon X5680
  - 6 core, 12 thread
- Our measurement was with single threaded G4
- New G4 is multi-threaded
- *We multiply by 12 to get CPU events / ms*

		20 MeV electron (pencil)	6 MeV photon (pencil)
CPU	Time (h)	29.44	12.42
	Events / ms / core	0.94	2.24
	Events / ms	*11.32*	*26.85*
GPU	Time (min)	22.23	9.65
	Events / ms	74.99	172.69
speedup (GPU/core)		79.49	77.19
speedup (GPU/CPU)		**6.62**	**6.43**

Other optimization experiments

- 6 MeV gamma pencil beam
- 1,000,000 events
- Voxels: 61 x 61 x 150

Experiment	Run time (s)	Events / ms
standard	9.33	107.2
prefer L1 cache	9.28	107.8
disable L1 cache	9.27	107.9
remove atomicAdd	9.31	107.4
limit to 32 registers	9.29	107.6
no thread divergence, no atomic add	**3.25**	**307.7**
no thread divergence, no atomic add, limit to 32 registers	**3.297**	**303.3**
add bounds checking	17.91	55.8

Going forward

- Hope to get factor 1.5 speed up from splitting particles into separate CUDA streams
  - Reduce thread divergence
  - Allow concurrent kernel launches
- Some possible benefits from using texture memory and hardware interpolation for lookup tables
- Look for other opportunities in expensive kernels

Acknowledgements

- NVidia for support of the CUDA Center of Excellence
- Koichi Murakami
- Makoto Asai, Joseph Pearl, Andrea Dotti @ SLAC
- Takashi Sasaki @ KEK
- Akinori Kimura, Ashikaga Institute of Technology