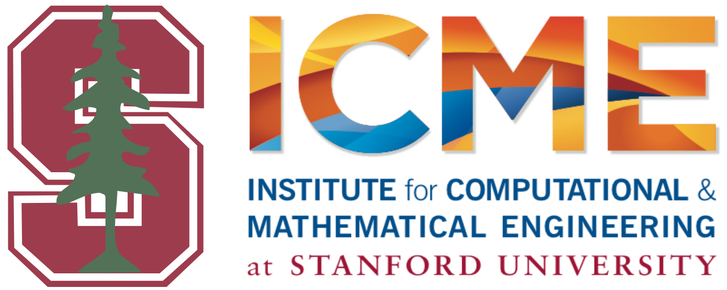
Dealing with Thread Divergence in a **GPU Monte Carlo** Radiation Therapy Simulator

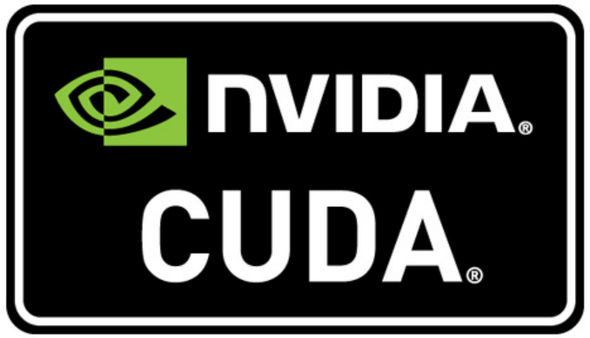
Nick Henderson, Stanford University

GPU Technology Conference 2015

Collaboration

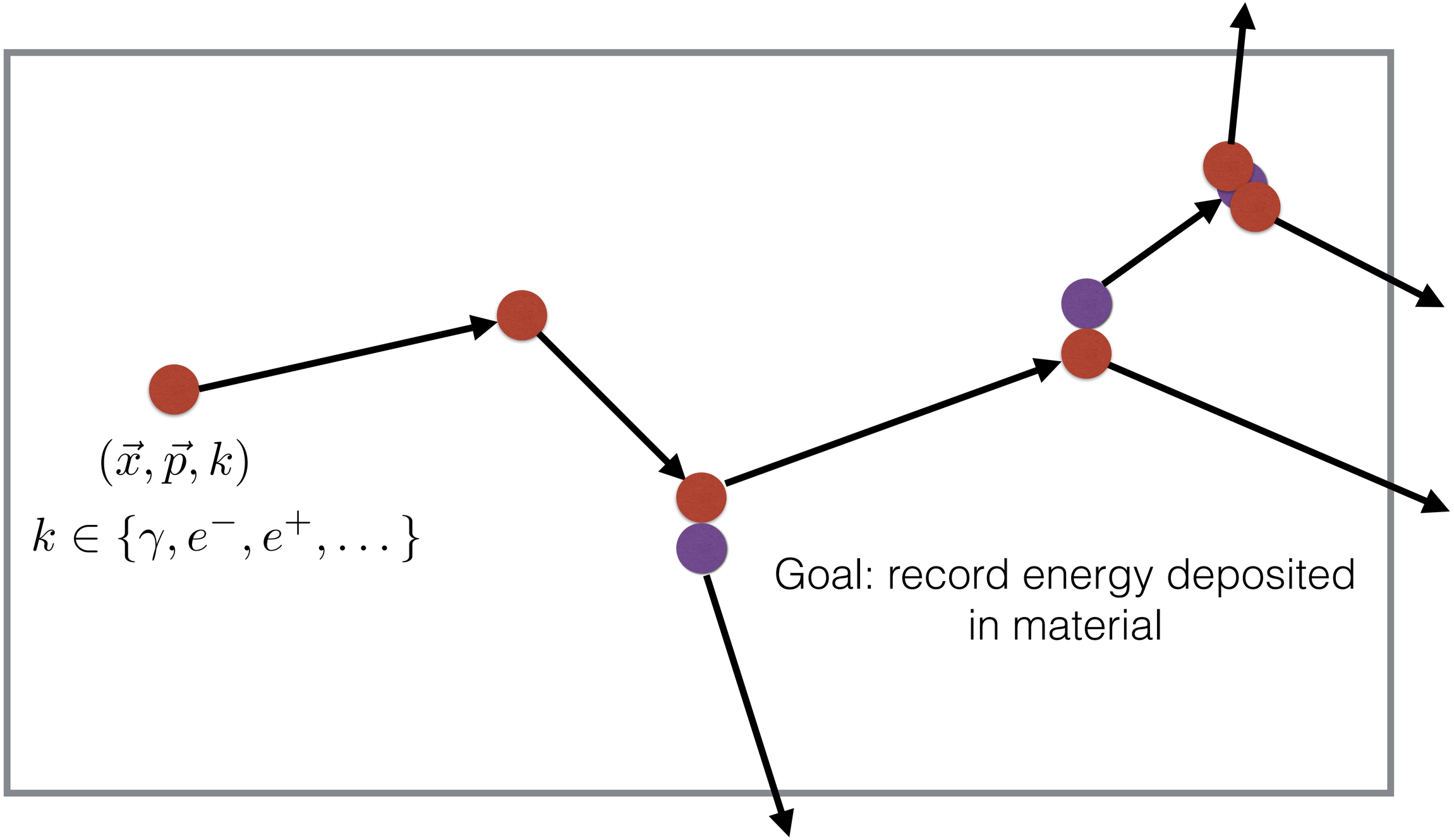
- Makoto Asai, SLAC
- Joseph Perl, SLAC
- Andrea Dotti, SLAC
- Takashi Sasaki, KEK
- Koichi Murakami, KEK
- Shogo Okada, KEK
- Akinori Kimura, Ashikaga Institute of Technology
- Margot Gerritsen, ICME
- Nick Henderson, ICME





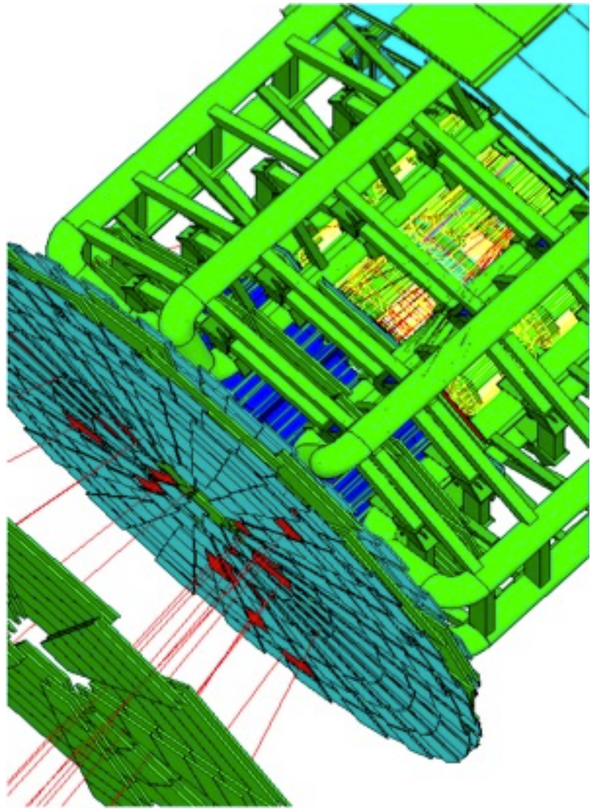
Special thanks to the CUDA Center of Excellence Program

Big picture


$(\vec{x}, \vec{p}, k)$
$k \in \{\gamma, e^-, e^+, \dots\}$
Goal: record energy deposited in material

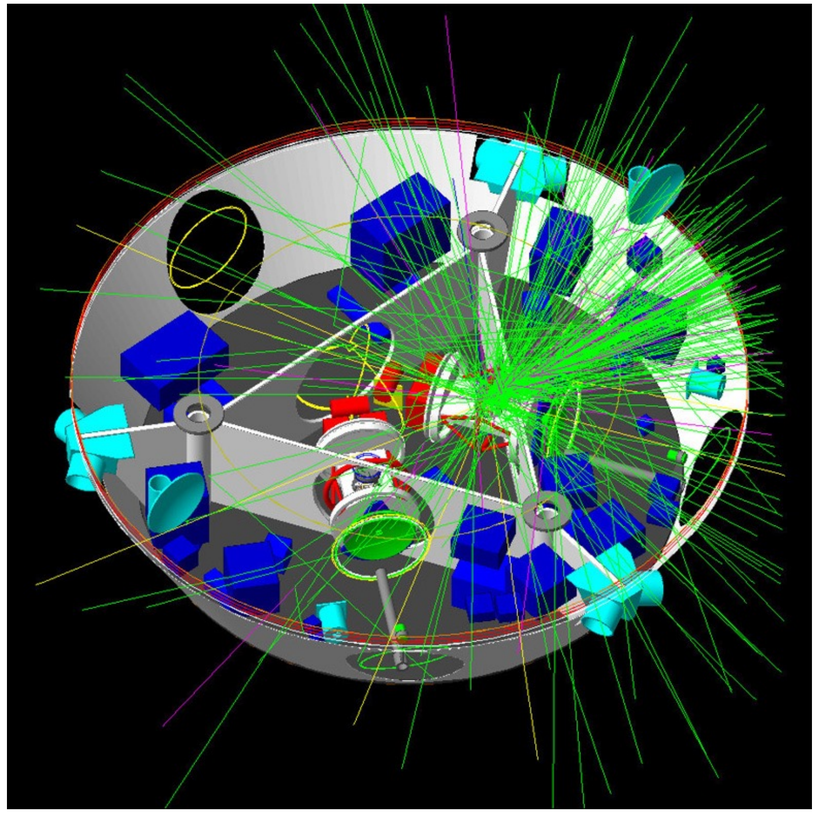
Geant4

High energy physics



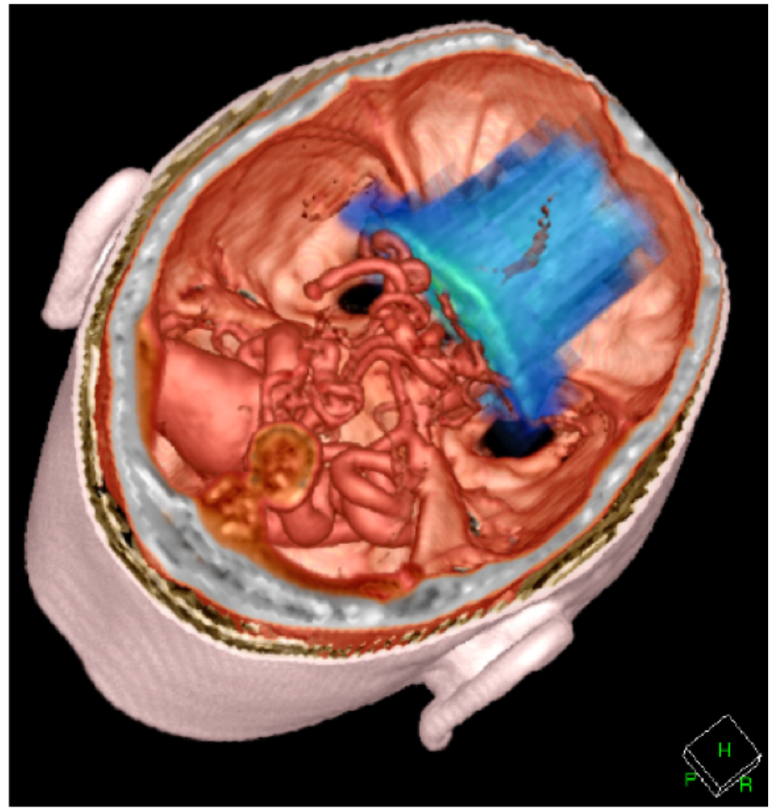
ATLAS

Space & radiation



LISA

Medical physics



gMocren

Images from Geant4 gallery and gMocren

Monte Carlo for X-ray radiotherapy simulation

Analytic methods

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

Monte Carlo methods

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

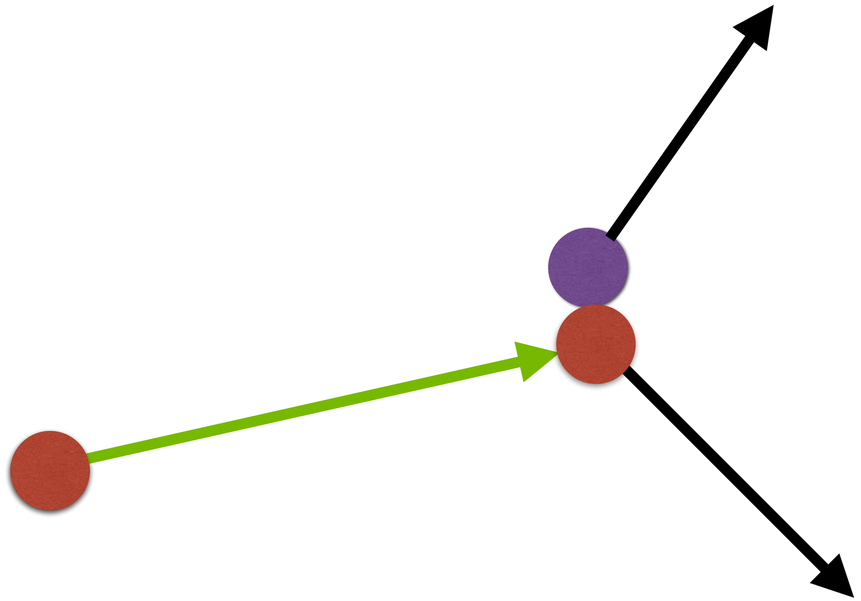
Good candidate for GPU implementation

- 3 particle kinds {ɣ,e-,e+}
- Low energy electromagnetic physics
- 1 material (H_2O)
- Uniformly discretized geometry

Monte Carlo Method

For all particles, repeat:

1. **Sample** step length & **limiting physics process**
2. Apply **physics processes** that occur **along** the step
3. **Sample physical interaction** that occurs at the end of the step



Implementation details

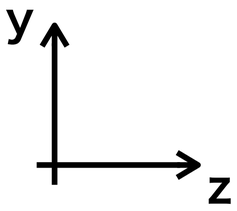
- Each **GPU** thread is responsible for an “active” particle
- Secondary particles are stored in thread local stacks
- Energy dose is stored in large global array and accumulated with `atomicAdd`

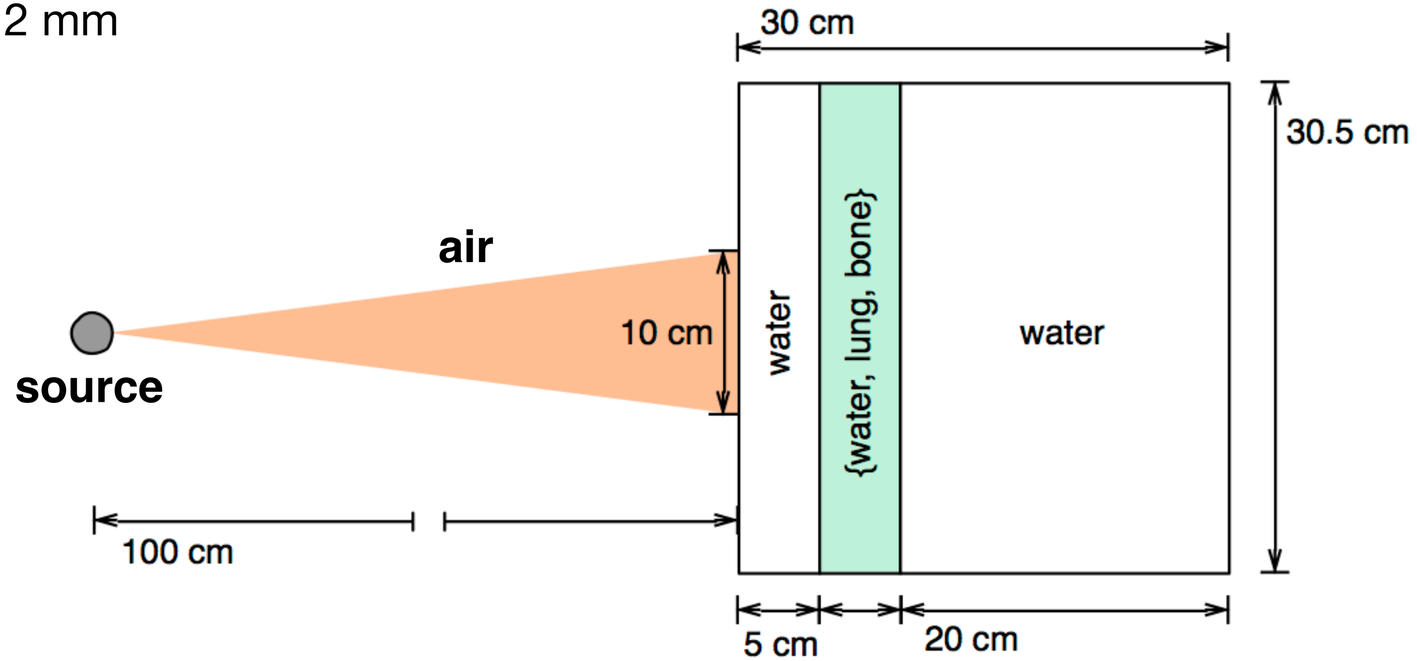
Performance and Validation

Verification for Dose Distribution

Dose Distribution of slab phantoms

- phantom size : 30.5 x 30.5 x 30 cm
- voxel size : 5 x 5 x 2 mm
- field size : 10 cm^2
- SSD : 100 cm
- slab materials :
  - (1) water
  - (2) lung
  - (3) bone





Beam particle and its initial kinetic energy:
- electron with 20MeV
- photon with 6MV Linac
- photon with 18MV Linac

	density
water	1.0 g/cm^3
lung	0.26 g/cm^3
bone	1.85 g/cm^3
air	0.0012 g/cm^3

blocks, # threads/block optimization: γ 6MV

- γ 6MV broad beam
- # of primaries: 32M photons
- table shows “run time/shortest time”
- voxels: 61 x 61 x 150
- 1.00 = 135.13 (sec)
- ~ 236 (primaries/msec)

	threads per block				
blocks	***32***	***64***	***128***	***256***	***512***
32	*32.26*	*17.27*	*9.68*	*5.34*	*3.21*
64	*17.27*	*9.69*	*5.34*	*3.17*	*2.20*
128	*9.71*	*5.34*	*3.09*	*2.13*	*1.58*
256	*5.88*	*3.34*	*2.04*	*1.49*	*1.29*
512	*3.89*	*2.22*	*1.45*	*1.17*	*1.24*
1024	*2.75*	*1.66*	*1.14*	*1.11*	*1.16*
2048	*2.24*	*1.39*	*1.08*	*1.02*	-
4096	*2.01*	*1.37*	***1.00***	-	-
8192	*2.08*	*1.29*	-	-	-
16384	*2.02*	-	-	-	-

blocks, # threads/block optimization: γ 18MV

- γ 18MV broad beam
- # of primaries: 32M photons
- table shows “run time/shortest time”
- voxels: 61 x 61 x 150
- 1.00 = 152.94 (sec)
- ~ 209 (primaries/msec)

	threads per block				
blocks	***32***	***64***	***128***	***256***	***512***
32	*31.59*	*16.95*	*10.28*	*5.44*	*3.22*
64	*16.96*	*10.21*	*5.45*	*3.18*	*2.22*
128	*10.20*	*5.46*	*3.14*	*2.11*	*1.65*
256	*6.01*	*3.45*	*2.06*	*1.48*	*1.35*
512	*3.88*	*2.24*	*1.44*	*1.18*	*1.21*
1024	*2.77*	*1.65*	*1.15*	*1.03*	*1.20*
2048	*2.26*	*1.40*	*1.01*	*1.02*	-
4096	*2.04*	*1.27*	***1.00***	-	-
8192	*1.93*	*1.29*	-	-	-
16384	*2.02*	-	-	-	-

blocks, # threads/block optimization: e- 20MeV

- e- 20MeV broad beam
- # of primaries: 32M electrons
- table shows “run time/shortest time”
- voxels: 61 x 61 x 150
- 1.00 = 285.01 (sec)
- ~ 112 (primaries/msec)

	threads per block				
blocks	***32***	***64***	***128***	***256***	***512***
32	*26.42*	*14.71*	*8.09*	*4.39*	*2.73*
64	*14.73*	*8.08*	*4.38*	*2.63*	*1.95*
128	*8.10*	*4.38*	*2.59*	*1.84*	*1.53*
256	*5.21*	*2.90*	*1.77*	*1.42*	*1.29*
512	*3.41*	*1.94*	*1.38*	*1.18*	*1.17*
1024	*2.54*	*1.56*	*1.14*	*1.05*	*1.15*
2048	*2.30*	*1.36*	*1.02*	*1.02*	-
4096	*2.13*	*1.26*	***1.00***	-	-
8192	*2.04*	*1.26*	-	-	-
16384	*2.09*	-	-	-	-

Computation Time Performance

185~250 times speedup against single-core G4 simulation!!

	e- beam with 20MeV		
	(1) water	(2) lung	(3) bone
G4 [msec/particle]	1.84	1.87	1.65
G4CU [msec/particle]	0.00881	0.00958	0.00885
× speedup factor (= G4 / G4CU)	***208***	***195***	***193***

GPU:

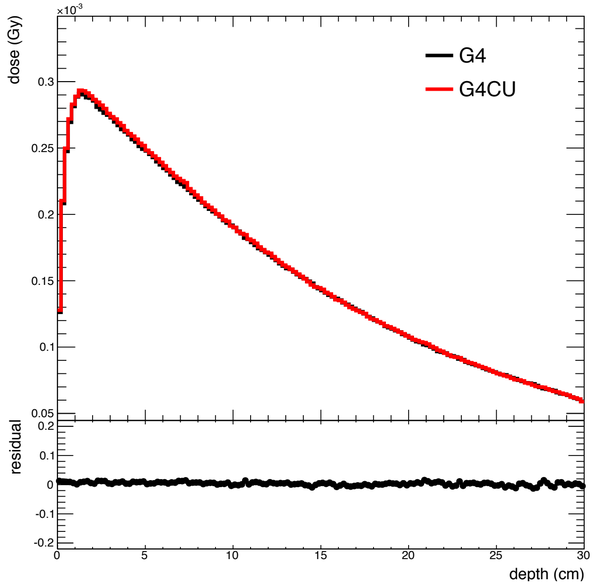
- Tesla K20c (Kepler architecture)
- 2496 cores, 706 MHz
- ***4096 x 128 threads***
- ***# of primaries***
  - ***50M particles -> e- 20MeV***
  - ***500M particles -> γ 6MV, 18MV***

CPU:
- Xeon E5-2643 v2 3.50 GHz

	γ beam with 6MV			γ beam with 18MV		
	(1) water	(2) lung	(3) bone	(1) water	(2) lung	(3) bone
G4 [msec/particle]	0.780	0.822	0.819	0.803	0.857	0.924
G4CU [msec/particle]	0.00336	0.00331	0.00341	0.00433	0.00425	0.00443
× speedup factor (= G4 / G4CU)	***232***	***248***	***240***	***185***	***201***	***208***

Comparison of depth dose for γ 6MV

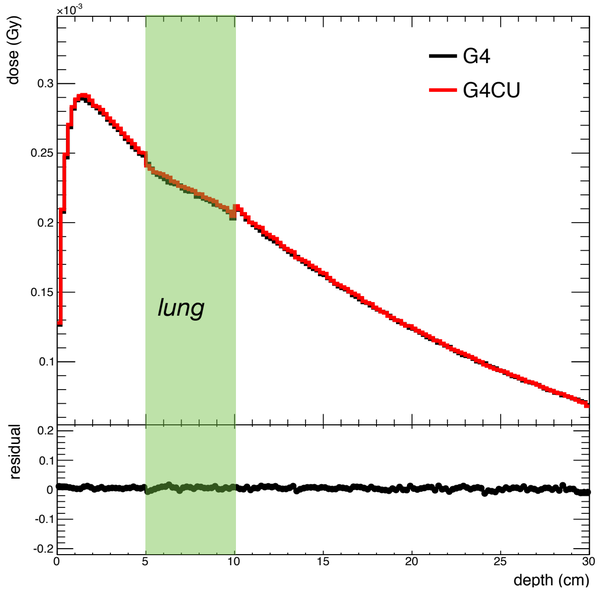
(1) water



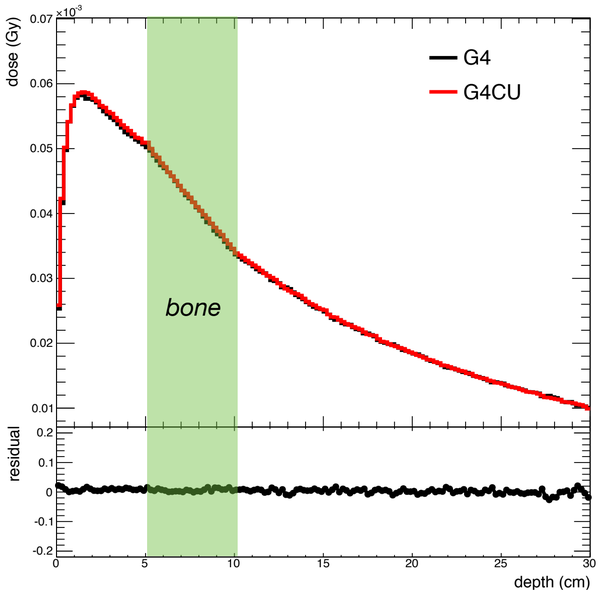
– G4 v9.6.3
– G4CU

- x-axis: z-direction (cm)
- y-axis: dose (Gy)
- residual = (G4CU–G4) / G4

(2) lung

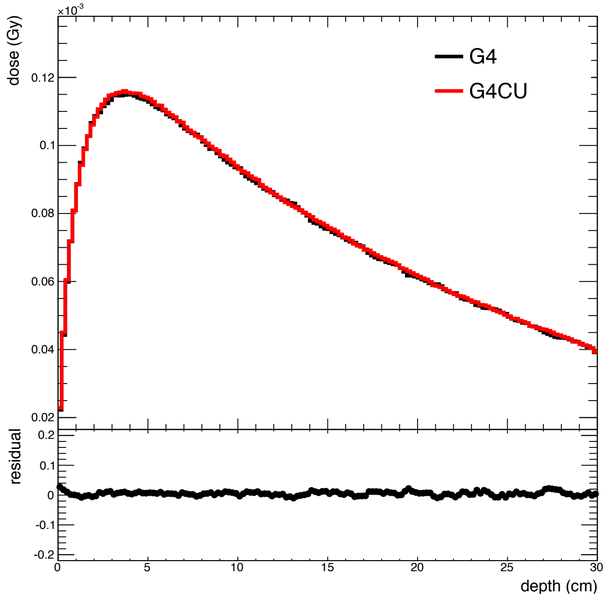


(3) bone



Comparison of depth dose for γ 18MV

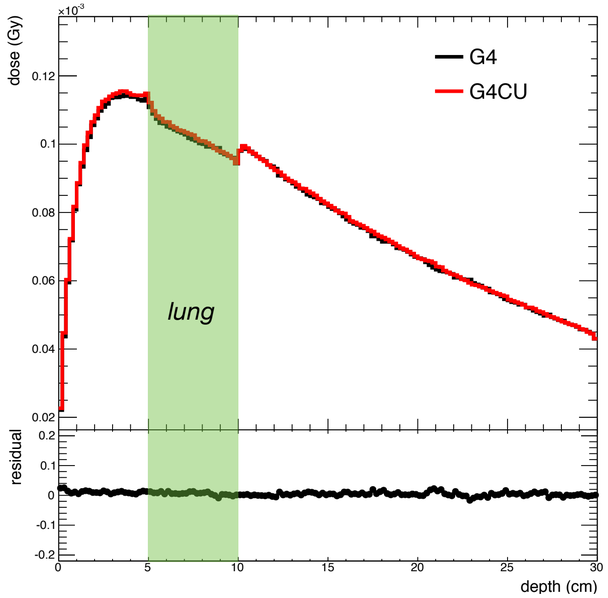
(1) water



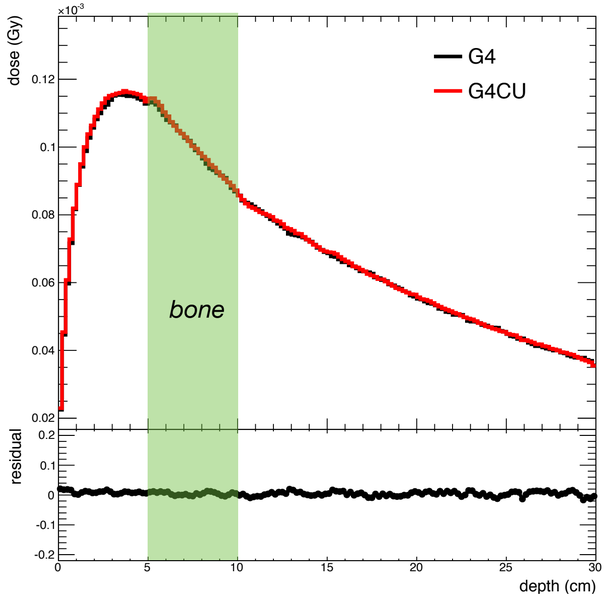
– G4 v9.6.3
– G4CU

- x-axis: z-direction (cm)
- y-axis: dose (Gy)
- residual = (G4CU–G4) / G4

(2) lung

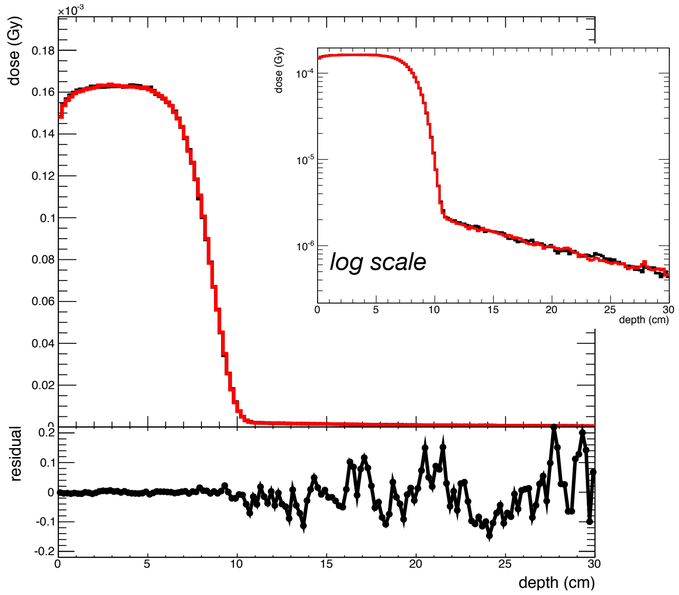


(3) bone



Comparison of depth dose for e- 20MeV

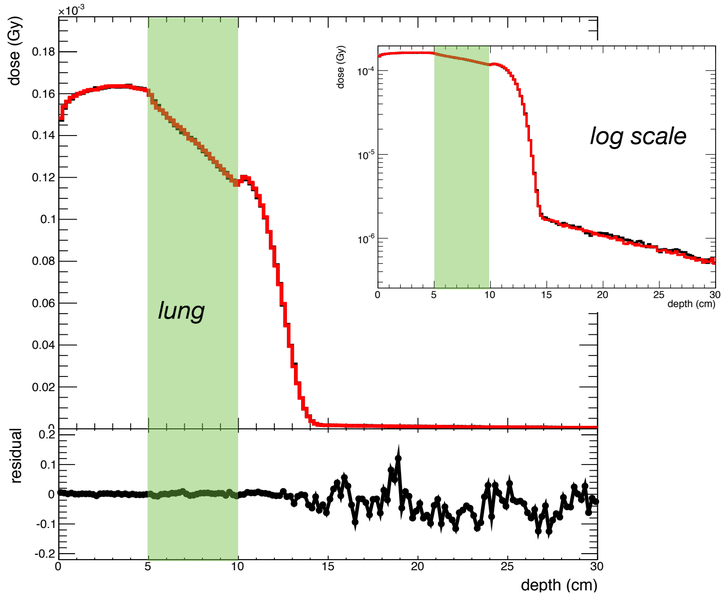
(1) water



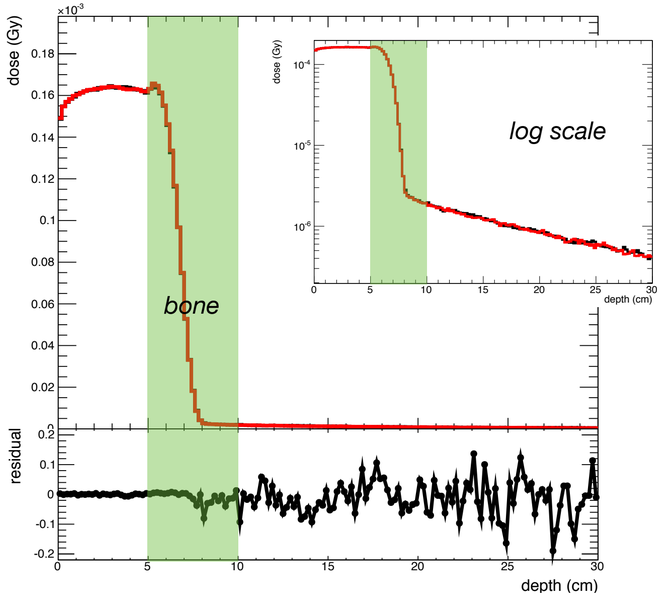
– G4 v9.6.3
– G4CU

- x-axis: z-direction (cm)
- y-axis: dose (Gy)
- residual = (G4CU–G4) / G4

(2) lung

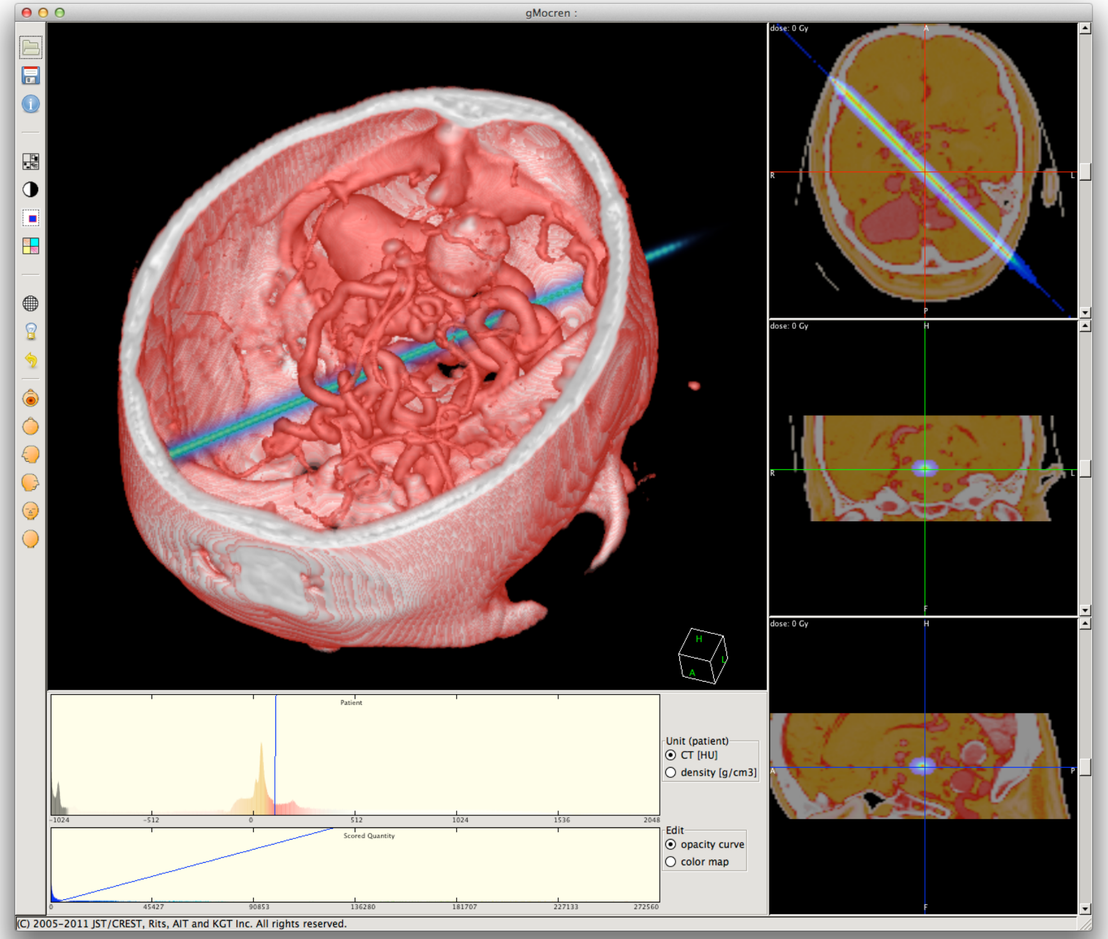


(3) bone

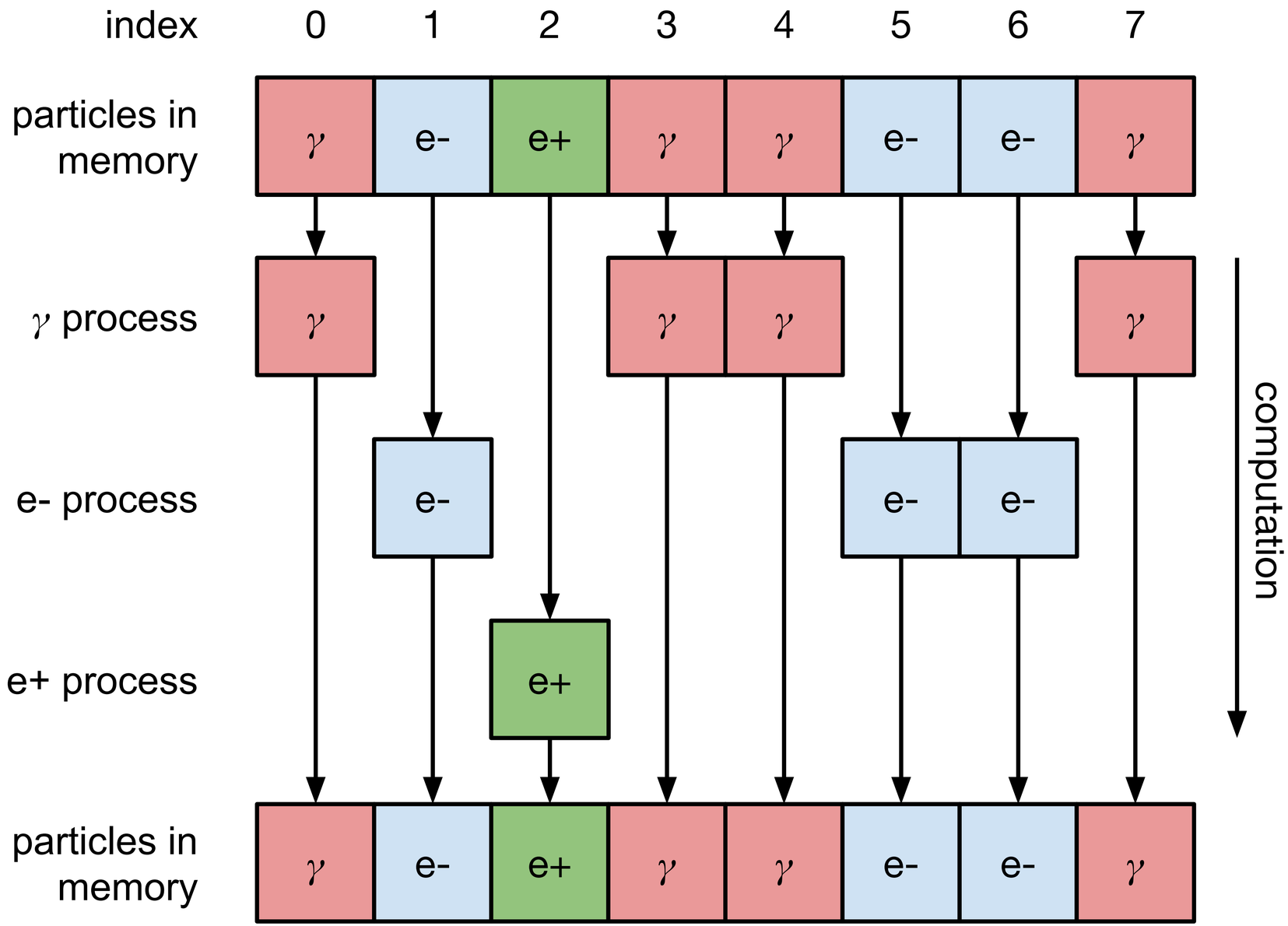


Visualization with gMocren

- Prototype integration with gMocren for visualization
- Pencil beam configuration
- Not an example of a treatment plan



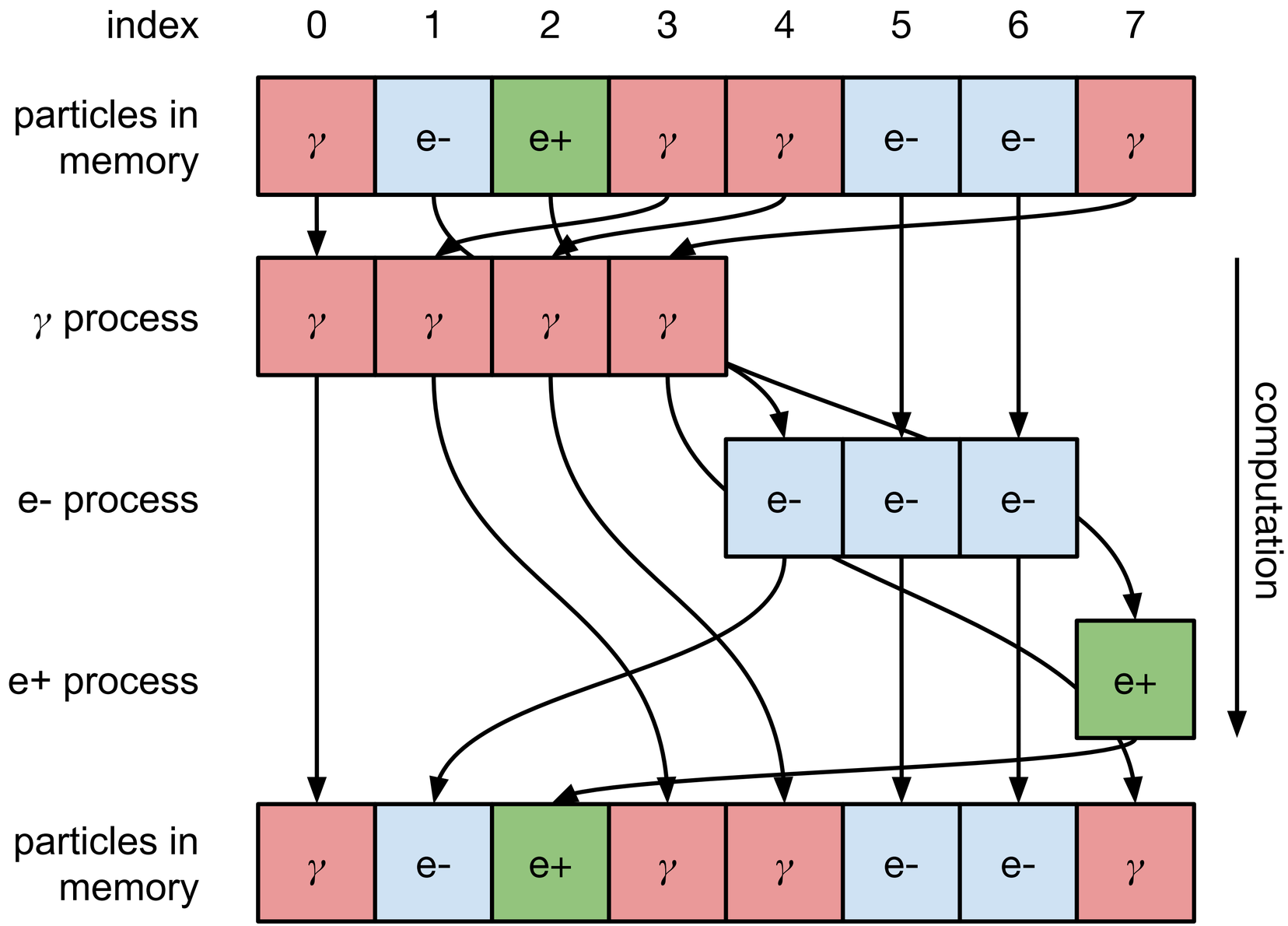
Dealing with Thread Divergence

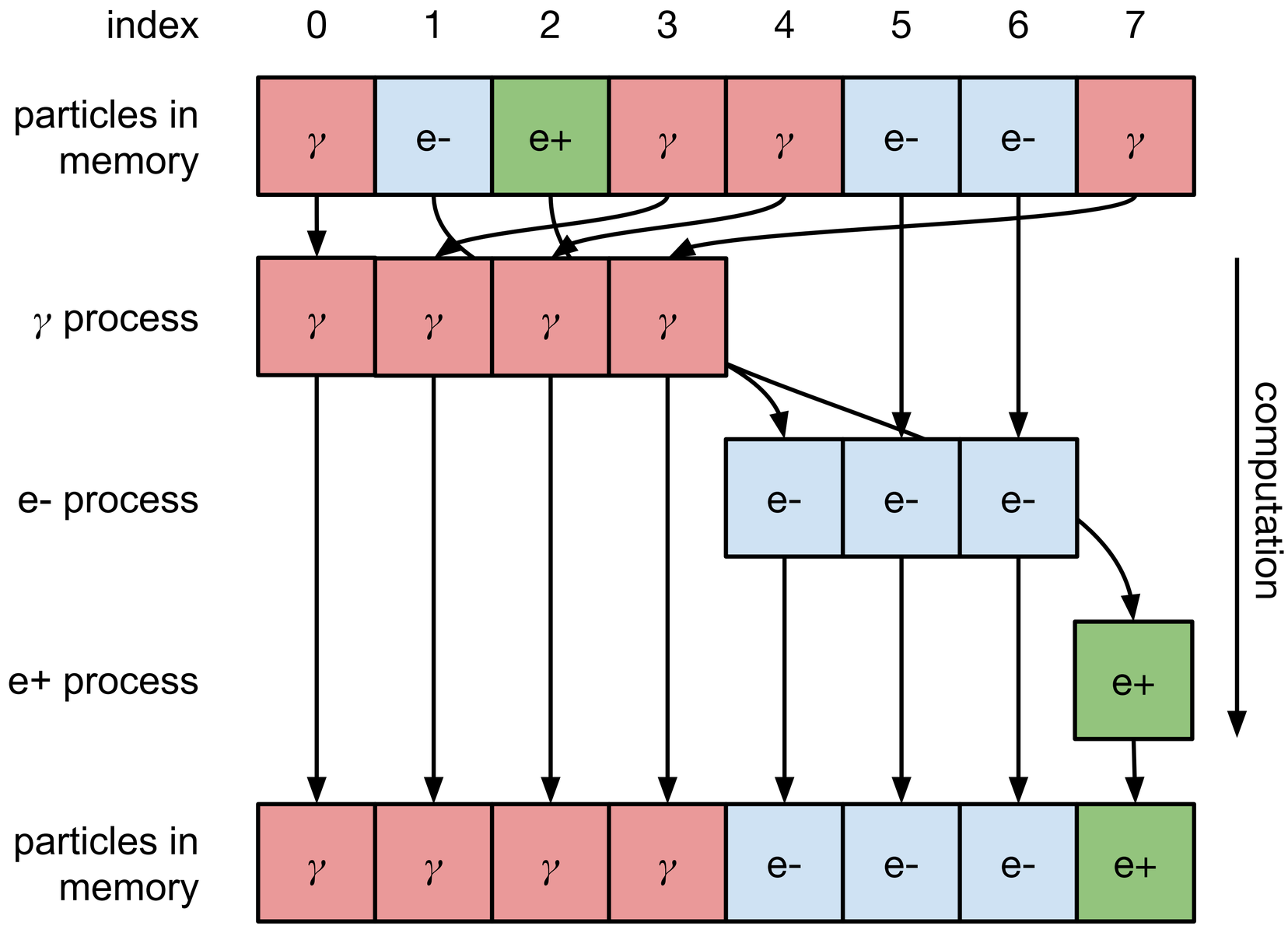

index
0
1
2
3
4
5
6
7
particles in memory
γ
e-
e+
γ
γ
e-
e-
γ
γ process
γ
γ
γ
γ
e- process
e-
e-
e-
e+ process
e+
particles in memory
γ
e-
e+
γ
γ
e-
e-
γ
computation

Experiment 1

- Initialize all threads to have same RNG seed
  - all threads will have same particle and select same physics process in each step
- Disable `atomicAdd` for global reduction
  - avoid serialization
- **Speedup: 3x** (~100 events/ms to ~300 events/ms)
  - no divergence, but non-physical

Ideas


index
0
1
2
3
4
5
6
7
particles in memory
γ
e-
e+
γ
γ
e-
e-
γ
γ process
γ
γ
γ
γ
e- process
e-
e-
e-
e+ process
e+
computation
particles in memory
γ
e-
e+
γ
γ
e-
e-
γ


index
0
1
2
3
4
5
6
7
particles in memory
γ
e-
e+
γ
γ
e-
e-
γ
γ process
γ
γ
γ
γ
e- process
e-
e-
e-
e+ process
e+
particles in memory
γ
γ
γ
γ
e-
e-
e-
e+
computation

Experiment 2

- Measure the time for a single simulation step with **131,072** active particles
  - **Step**: 5.2 ms
- Measure the time for a **sort** followed by a **run-length-encode** for **131,072** keys
  - **Thrust**: 1.1 ms (version 1.8.0)
  - **CUB**: 0.5 ms (version 1.3.2)

Simulation surrogate: autoregressive model

Autoregressive model

- Let each thread generate an independent autoregressive sequence
- Key performance parameter: number of time steps

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \cdots + \alpha_n x_{t-n} + \epsilon_t$$

$$x_t = \sum_{i=1}^{n} \alpha_i x_{t-i} + \epsilon_t$$

$$\epsilon_t \sim N(0, 1)$$

Extended autoregressive model

- In each time step, randomly select one of ***m*** different autoregressive models (a process)
- An autoregressive model is characterized by its set of coefficients
- Key performance parameter: number of processes (AR models)

$$x_t = \alpha_{1,p} x_{t-1} + \alpha_{2,p} x_{t-2} + \cdots + \alpha_{n,p} x_{t-n} + \epsilon_t = \sum_{i=1}^{n} \alpha_{i,p} x_{t-i} + \epsilon_t$$

$$\epsilon_t \sim N(0,1),\ p \sim U\{1,m\}$$

Calibration of surrogate

- Number of processes: 9
  - 3 particle kinds and 3 processes per particle
- Number of time steps: 16
  - Evaluated experimentally
  - Intuitive match: 10 numbers per particle, 6 extra numbers for the physics process
- Result: ~ 5 ms per step
- Run with 262,144 active threads

Surrogate model performance

Time per thread step	1 kernel stream	1 stream per process
Original	18.9 ns	18.6 ns
Sort by process	8.70 ns	8.45 ns

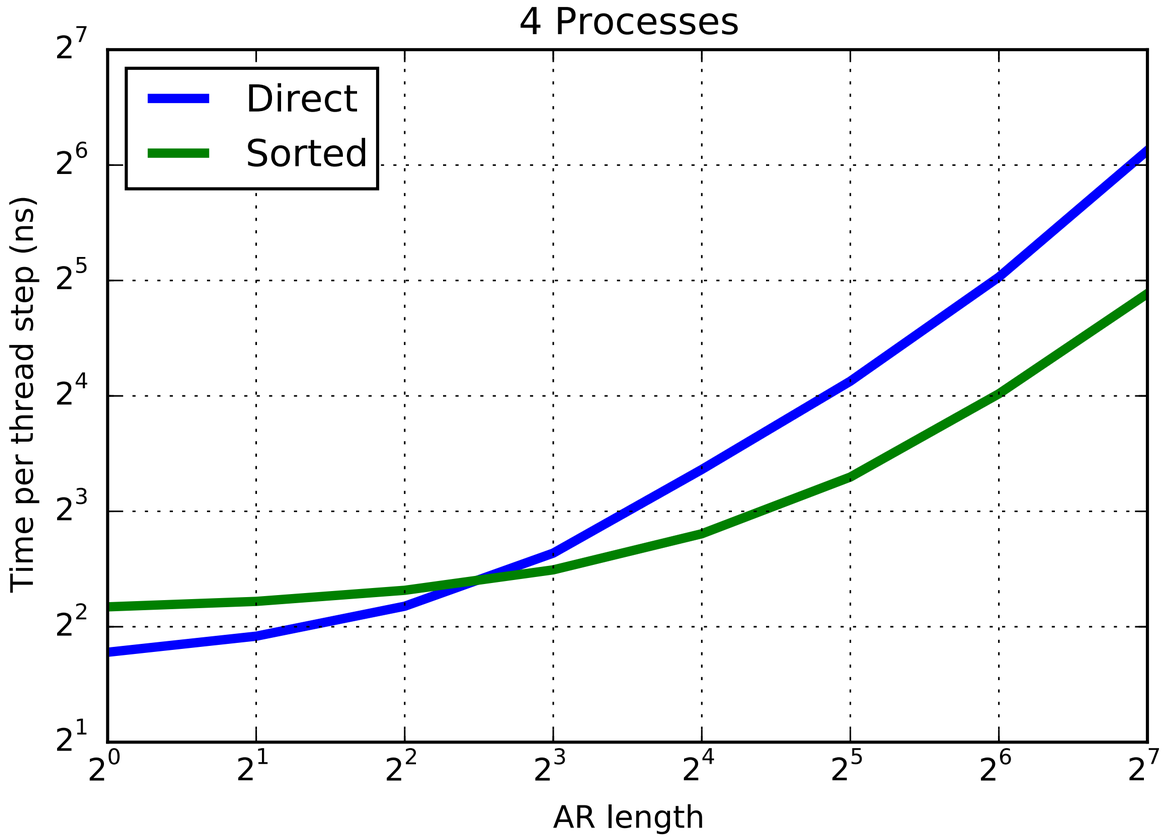
Surrogate model speedup

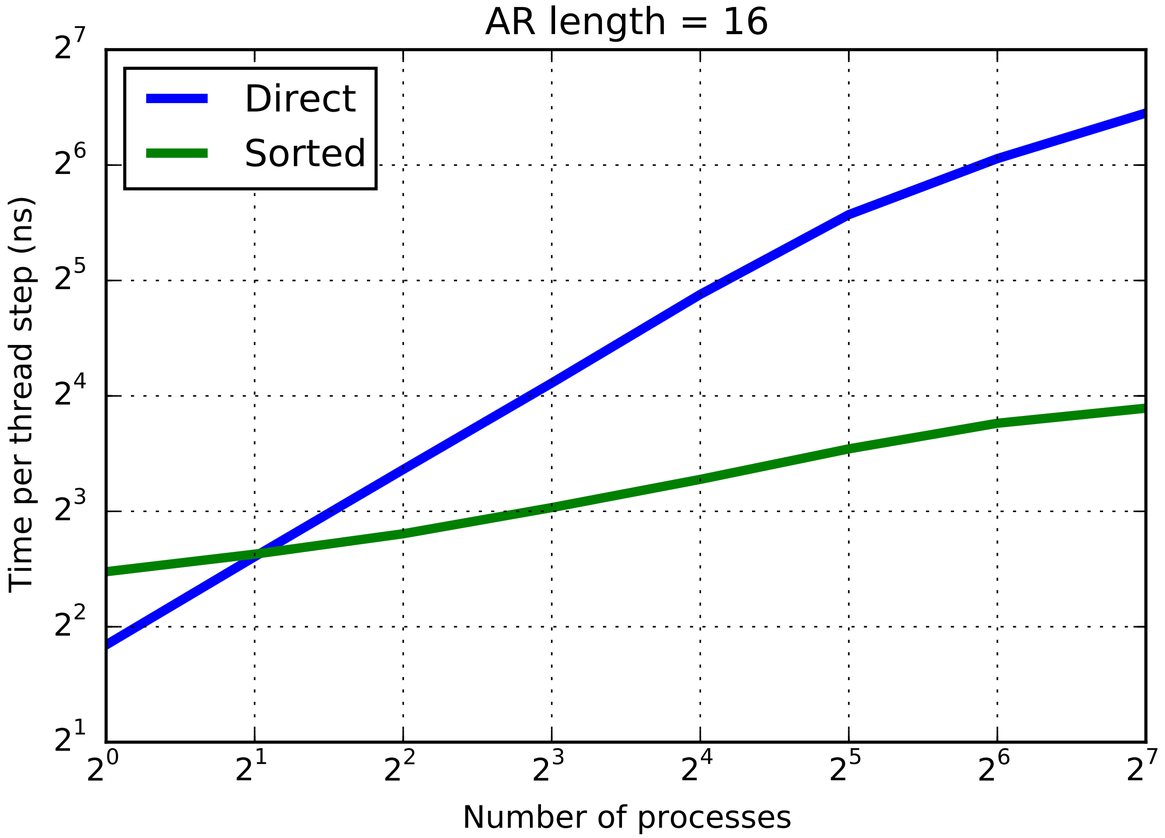
x	1 kernel stream	1 stream per process
Original	1.0	1.01
Sort by process	2.17	2.24

Speedup: # of processes v AR length

AR length \ Number of processes	1	2	4	8	16	32	64	128
1	0.53	0.65	0.76	0.95	1.31	1.84	2.68	4.07
2	0.54	0.67	0.81	1.04	1.45	2.07	2.93	4.23
4	0.55	0.72	0.91	1.22	1.82	2.60	3.44	4.64
8	0.58	0.81	1.11	1.60	2.39	3.34	4.21	5.25
16	0.64	0.98	1.47	2.11	3.04	4.08	4.90	5.89
32	0.73	1.18	1.78	2.57	3.57	4.61	5.39	6.34
64	0.82	1.37	2.02	2.83	3.90	4.95	5.71	6.61
128	0.89	1.66	2.36	3.10	4.05	5.11	5.87	6.74

Time per thread step





Conclusions

- Thread divergence is a key performance limiter for **Monte Carlo** methods with process selection
- For simulation of radiotherapy
  - Surrogate model suggests **2x** speedup with sorting strategy
- For general **Monte Carlo** method with process selection
  - Speedup with sorting grows with model complexity (see chart)

Thanks

- I am Nick Henderson
- You can email me: nwh@stanford.edu
- Read more about:
  - Geant4: http://geant4.cern.ch/
  - Our GPU simulator: http://dx.doi.org/10.1051/snamc/201404204
- Please fill out the survey!