Parallelization for Fast Image Reconstruction using the Stochastic Origin Ensemble Method for Proton Beam Therapy

UMBC REU Site: Interdisciplinary Program in High Performance Computing

Fernando X. Avila-Soto[1], Alec N. Beri[2], Eric Valenzuela[3], Abenezer Wudenhe[4],

RAs: Ari Rapkin Blenkhorn[4], Jonathan S. Graf[5], Samuel Khuvis[5], Mentor: Matthias K. Gobbert[5],

Client: Jerimy Polf, Department of Radiation Oncology, UMD School of Medicine

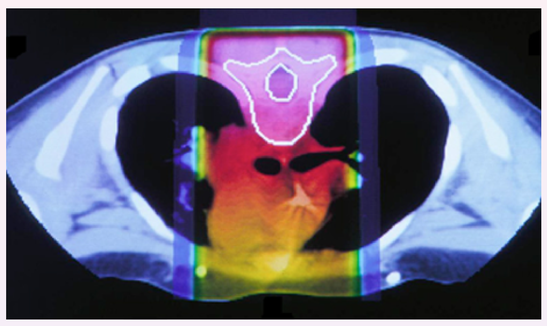
[1]Muskingum University, [2]UMD College Park, [3]CSU Channel Islands, [4]CSEE, UMBC, [5]Math & Stat, UMBC

Objective

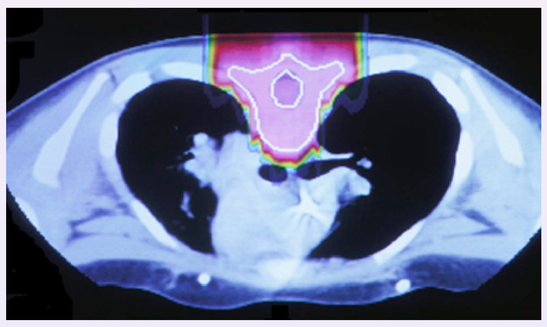
We apply parallel computing techniques to the Stochastic Origin Ensemble (SOE) algorithm for the reconstruction of images of secondary gammas emitted during proton beam therapy. We use OpenMP, CUDA, and Intel Compilers to optimize the C++ implementation of the algorithm on CPUs and a single GPU.

Proton Beam Therapy

The use of proton therapy for treating cancer has greatly increased over the past decade because of the advantageous properties of proton beams.



X-Ray Treatment



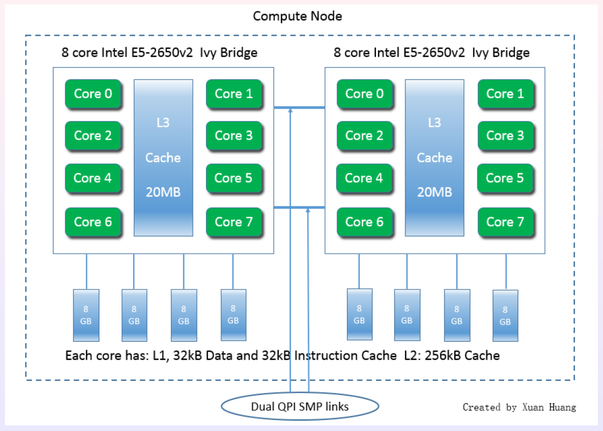
Proton Beam Treatment

Because of the uncertainties in the exact position of the distal dose gradient within the patient, a method of verifying the in vivo beam range is critcal.

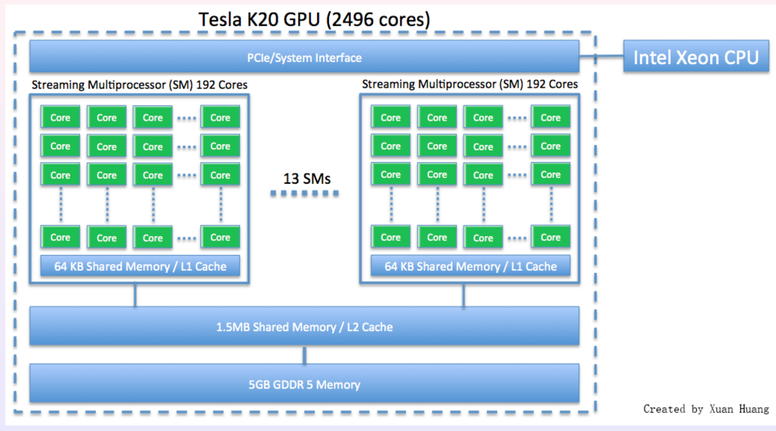
SOE algorithm is an effective method for reconstructing images of a proton pencil beam from data collected using an ideal Compton camera.

With a verification system, we can improve our ability to fully exploit the advantages of proton radiation therapy.

UMBC High Performance Computing Facility (`hpcf.umbc.edu`)



One node with two 8-core CPUs



One NVIDIA K20 GPU with 2496 cores

One Node with Two Eight-Core Intel E5-2650v2 Ivy Bridge CPUs

To test the SOE-based reconstruction code, we use a **small test case** with $\approx 15{,}000$ and a **large test case** with $\approx 380{,}000$ Prompt Gammas, with ≈ 40 mm^3 voxel size and 1,000 iterations. We used an OpenMP multi-threading approach compiled with GNU.

Test case	1 thread	2 threads	4 threads	8 threads	16 threads	32 threads
Wall clock time W_t in seconds						
Small	40	22	13	8	6	6
Large	955	513	286	163	101	108
Speedup $S_t = W_1/W_t$						
Small	1.00	1.82	3.08	5.00	6.67	6.67
Large	1.00	1.86	3.34	5.86	9.45	8.84

- Very good speedup on one node. Not effective to use more threads than CPU cores.

One NVIDIA K20 GPU with 2496 Computational Cores

NVIDIA CUDA (Compute Unified Device Architecture) for multi-threading on the GPU.

Test case	512 threads	1024 threads	2048 threads	4096 threads	8192 threads
Wall clock time W_t in seconds					
Small	93	60	38	25	15
Large	2061	1245	660	382	229
Speedup $S_t = W_1/W_t$					
Small	1.00	1.55	2.45	3.72	6.20
Large	1.00	1.66	3.13	5.40	9.00

- **Code successfully ported to hybrid CPU/GPU implementation!**
- Very good speedup on GPU. Very effective to use more threads than cores.

References and Acknowledgments

- Mackin, Peterson, Beddar, Polf, *Phys. Med. Biol.*, 2012
- Full technical report: HPCF–2015–27, `hpcf.umbc.edu` > Publications

REU Site: `hpcreu.umbc.edu`; NSF, NSA, DOD, UMBC, HPCF, CIRC