DEVELOPMENT OF GPU-BASED MONTE CARLO CODE FOR FAST CT IMAGING DOSE CALCULATION ON CUDA FERMI ARCHITECTURE

Tianyu Liu, Xining Du, Wei Ji and Xie George Xu*
Nuclear Engineering Program
Rensselaer Polytechnic Institute
Troy, New York 12180, USA
∗Corresponding author: xug2@rpi.edu

ABSTRACT

This paper describes the development of a Graphics Processing Unit (GPU) accelerated Monte Carlo photon transport code, $\mathrm{ARCHER_{GPU}}$, to perform CT imaging dose calculations with good accuracy and performance. The code simulates interactions of photons with heterogeneous materials. It contains a detailed CT scanner model and a family of patient phantoms. Several techniques are used to optimize the code for the GPU architecture. In the accuracy and performance test, a 142 kg adult male phantom was selected, and the CT scan protocol involved a whole-body axial scan, 20-mm x-ray beam collimation, 120 kVp and a pitch of 1. A total of 9×10^8 photons were simulated and the absorbed doses to 28 radiosensitive organs/tissues were calculated. The average percentage difference of the results obtained by the general-purpose production code MCNPX and $\mathrm{ARCHER_{GPU}}$ was found to be less than 0.38%, indicating an excellent agreement. The total computation time was found to be 8,689, 139 and 56 minutes for MCNPX, $\mathrm{ARCHER_{CPU}}$ (6-core) and $\mathrm{ARCHER_{GPU}}$, respectively, indicating a decent speedup. Under a recent grant funding from the NIH, the project aims at developing a Monte Carlo code with the capability of sub-minute CT organ dose calculations.

Key Words: GPU, CUDA, Monte Carlo CT dose calculation

1. INTRODUCTION

Clinical use of x-ray Computed Tomography (CT) — an important diagnostic imaging tool — has continued to grow by a factor of 10% to 15% per year [1]. This trend has caused a significant increase in the radiation dose delivered to the patient population, leading to a mounting concern over the public health [2]. The radiology community has called for better tools for quantifying and reporting patient-specific dose for CT procedures [3]. Monte Carlo simulation is one of the common methods used in radiation dosimetry study. Well known for its high accuracy, it allows exact modeling of 3D geometries, includes precise mathematical models for radiation particle interactions with matter, and adopts a cross-section representation of the physics that is as accurate as theory and experiment permit [4]. Nonetheless, the computation time to obtain a dose estimate with an acceptable statistical uncertainty is prohibitively long, making a routine use impractical in a clinical setting where the computing resource is usually limited to a desktop. One feasible solution is to employ the Graphics Processing Unit (GPU), a rapidly developing paradigm for heterogeneous parallel computing. While the serial code may still run on the CPU-based host machine, the parallelizable, compute-intensive kernels can be offloaded through the PCIe interface to the plug-and-play GPU device, which has the advantage of high energy efficiency [5]. By November 2012, 53 out of the top 500 supercomputers worldwide have utilized GPUs as the accelerator to boost the overall system performance [6]. Recently, several research groups have adopted the GPU technology to medical applications and have reported impressive speedup factors [7,8,9]. However, a comprehensive tool to

quantify individual organ doses using an accurate physics model, a validated radiation source as well as a representative collection of patient models is still absent. In this paper, we report our effort in developing a new Monte Carlo photon transport code, called "ARCHER" (version 0.9.1), a testbed for Accelerated Radiation-transport Computation in Heterogeneous EnviRonments, for accurate and fast CT dose calculations. The code consists of two variants, $\mathrm{ARCHER_{CPU}}$ and $\mathrm{ARCHER_{GPU}}$ intended for a fair CPU-GPU performance comparison (In the rest of the paper, the name without a subscript refers to both code variants). Besides, the code includes a detailed CT scanner model and a family of computational human models, which were extensively validated in previous works [10,11,12,13].

2. MATERIALS AND METHODS

2.1. Development of Monte Carlo Photon Transport Code

2.1.1. Physics model

ARCHER is developed to simulate 1keV∼20MeV photon transport in highly heterogeneous media. For the CT dosimetry application, the energy range of interest is set to 1∼140keV, where photoelectric effect, incoherent and coherent scattering may occur. For the scattering models, to account for electron's binding effects, incoherent and coherent form factors are used to modify the angular distribution that is initially sampled from the classical Klein-Nishina and Thomson formulae [14,15]. All the interactions are kept at the atomic scale, with the attenuation coefficients being calculated on the fly from the raw photo-atomic cross-sections using logarithmic interpolation, according to the elemental composition of the material. Two techniques are used to accelerate the simulation process. One is the Kahn's sampling method which increases the efficiency of rejection sampling for incoherent scattering simulation [16]; the other is the delta tracking method which speeds up the path-length sampling in the lattice geometry [17]. Because the Continuous Slowing Down Approximation (CSDA) range of secondary electrons inside the human body at the considered energy range is generally one order of magnitude smaller than the dimension of a voxel [18], their transport process is not simulated in ARCHER and their energy is deemed to be locally deposited.

2.1.2. Dose Calculation

Dose calculation is performed through a collision estimator according to Eq. 1, where m is the mass of the tallied organ, H is the heating number representing the average energy deposition per collision, Σ_t is the total linear attenuation coefficient, and ψ is the angular flux of the particle.

$$D = \frac{1}{m} \int dE \int dt \int dV \int d\Omega H(E) \Sigma_t(\vec{r}, E) \psi(\vec{r}, \vec{\Omega}, E, t) \tag{1}$$

A special treatment is made to two body structures listed in ICRP recommendations [19]. First, the dose to bone surfaces is approximated by the dose to spongiosa [13,20]. Second, the dose to red bone marrow is tallied by a variation of the collision estimator, analytically expressed by Eq. 2, where R_{RBM} is the dose response function serving as an energy-dependent weighting factor to the flux [13].

$$D_{RBM} = \frac{1}{m} \int dE \int dt \int dV \int d\Omega R_{RBM}(E) \psi(\vec{r}, \vec{\Omega}, E, t) \tag{2}$$

2.2. Computational Human Phantom Model

ARCHER contains ten extended RPI-Adult males and females, a family of voxel human phantoms representing patients of various weights from normal to overweight and to morbidly obese [10], as is shown in Fig. 1. The phantoms are composed of $0.35 \times 0.35 \times 0.35\ mm^3$ voxels and have over 100 organs or tissues defined in conformity with the ICRP reference anatomical data [19]. The obesity level is reflected by the different amount of subcutaneous and visceral adipose tissues added to the phantom, which were found to have remarkable shielding effect to low energy x-ray [10]. In ARCHER, the phantoms are stored as external text files and loaded at runtime.

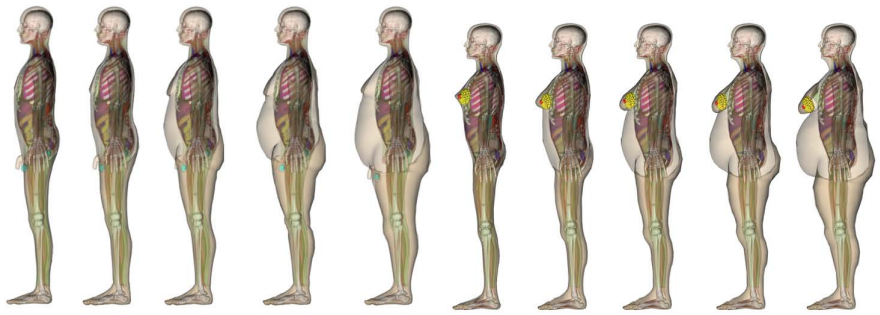

Figure 1. Extended RPI-Adult male and female computational phantoms.

2.3. CT Scanner Model

The CT scanner model was based on a GE LightSpeed third-generation 16-multi-detector CT (MDCT) shown in Fig. 2. The source photons are emitted from the anode of the x-ray tube in a shape of a fan beam and subtended by a flat filter. The energy distribution of the source is obtained from xcomp5r [21], a software that generates x-ray spectra. The bowtie filter is defined by the union of a parallelepiped and two elliptic cylinders. The geometric parameters are determined through an iterative trial-and-error method until the simulation result converges to experimental CTDI values [12]. In ARCHER, the CT scanner model is hard-coded by translating the original MCNPX-style text format into C language. A series of scan protocols are predefined, including a combination of scan mode (helical or axial), beam collimation (5, 10, or 20 mm) and kVp (80, 100, 120 or 140).

2.4. The Architecture of CUDA-enabled GPU[22]

Compute Unified Device Architecture (CUDA) is a general purpose parallel computing platform and a programming model introduced by Nvidia. It includes an evolutionary series of hardware architectures such as Fermi and Kepler. A GPU typically contains hundreds or thousands of Streaming Processors (SPs), leading to a considerable arithmetic throughput on the order of TeraFLOPS.

CUDA also includes software support and enables developers to write code in the extensions of high-level languages such as C and Fortran with great convenience. Specifically, the parallelizable part of

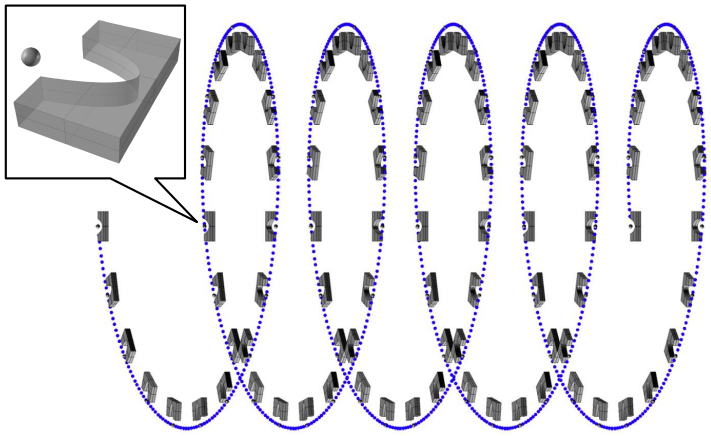

Figure 2. Trajectory of the CT scanner model in a helical scan. For illustration purpose, a large pitch value is selected.

$\mathrm{ARCHER_{GPU}}$ is wrapped into a special function called ***kernel*** and is isolated from the rest of the code that runs in sequence on the CPU side (called host). The kernel is then executed in parallel on the GPU side (called device) by a user-specified number of ***threads***. Here two points are noteworthy. One is that the number of threads can be very large but not all of them are able to be launched at the same time. In fact the GPU threads are further evenly grouped into a coarser unit called ***blocks***, which are delivered to SPs batch by batch in sequence. The number of blocks that each batch can carry reflects the actual number of threads that can be physically running on the GPU at a time, and is limited by the hardware capability. Properly specifying the number of threads per block helps maximize hardware usage and achieve a performance sweet spot. The other point is that threads (the software execution unit) and SPs (the hardware computation engine) do not have one-to-one mapping. Instructions from a large number of different threads are generally pushed into the pipelines of a small fixed number of SPs for concurrent execution.

2.5. Code Implementation on GPUs

2.5.1. Code design

$\mathrm{ARCHER_{GPU}}$ is designed to run on our CPU-GPU heterogeneous system, consisting of an Intel Xeon 5650 6-core CPU, 13 GB RAM, and a Fermi-based Nvidia Tesla M2090 GPU with 512 streaming processors (SP) and 6 GB RAM. The central part of $\mathrm{ARCHER_{GPU}}$, i.e. simulation of photon transport is written in CUDA C and performed in parallel on the device, whereas the preceding and following parts, including file I/O, device configuration, etc, are written in C and executed in sequence on the host. The diagram is shown in Fig. 3.

The parallel code includes two parts, a pseudo-random number initialization (PRNI) kernel and a particle tracking (PTK) kernel. The PRNI kernel generates a large batch of random number states and is launched only once through the whole application. The algorithm used is XORWOW [23] provided by the CURAND library. The PTK kernel tracks the history of photons and records dose information. It is launched per scanner rotation and reuses the initialized random number states produced by the PRNI kernel. In PTK, each thread is empirically assigned 100 photon histories to simulate. This has two benefits compared to assigning only one photon to one thread. First, the execution time of PRNI is reduced to

1/100, since it is proportional to the number of random number seeds, which is equal to the number of threads launched in PTK. Second, it decreases the thread launching overhead. We use the occupancy calculator, an excel spreadsheet provided by Nvidia, to determine the optimal number of threads per block, which is 256 in this study. The total number of blocks required is then collectively determined by the number of histories per thread, the number of threads per block and the total number of histories to be simulated. At runtime, the GPU hardware implicitly schedules the block and thread assignment to the SPs, as is described in Section 2.4. Consequently, there is no need to manage the block- or thread-level iterations manually. Another issue related to the PTK design is that the conditional statements inherent in the conventional history-based Monte Carlo method inevitably cause branch divergence, resulting in different branches being executed sequentially. Although this problem is not dealt with in this paper, it can be potentially solved by using the event-based vectorized Monte Carlo algorithm.

In addition, $\mathrm{ARCHER_{CPU}}$ is written in C with MPI-OpenMP and designed to run on the multi-core CPU only with the capability of utilizing all the available cores. To make the performance comparison as fair as possible, the same random number generator, physics model and similar compiler options are adopted for both $\mathrm{ARCHER_{CPU}}$ and $\mathrm{ARCHER_{GPU}}$.

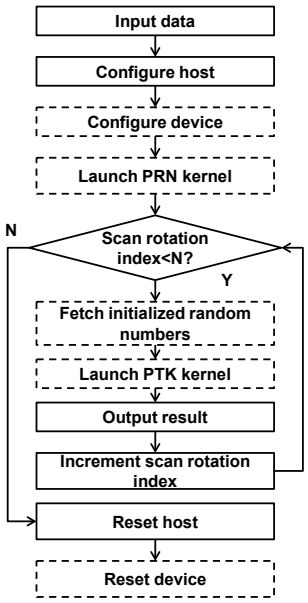

Figure 3. Flowchart of $\mathrm{ARCHER_{GPU}}$. The dashed boxes represent functions executed on the GPU device.

2.5.2. Memory management[24,25]

An important issue to address in PTK is the optimization of memory usage. Memory layout on a GPU is illustrated in Fig. 4. The on-chip register is used to store per-thread temporary variables such as particle attributes and local random number state that are continually updated. The off-chip DRAM on GPUs consists of three different types: user allocated space, statically allocated space and runtime heap. Firstly, the user allocated space is often termed as global memory and in our case is used to store data of large size. There are two patterns of data access to the global memory in $\mathrm{ARCHER_{GPU}}$. One is a scattered, random

pattern. Some data such as phantoms and cross-sections have to be fetched in this pattern due to the stochastic nature of Monte Carlo methods. This typically results in an uncoalesced access and can hardly be modified. We mostly rely on the L1 and L2 cache on the GPUs to mitigate the problem. All the other data such as global random number states are accessed in a memory-favored coalesced manner such that threads with contiguous indices always access memories with contiguous addresses. It is worth mentioning that the random number states reside in two types of memory space: at first, they have to be allocated and initialized in the global memory in the PRNI kernel. Then at the beginning of the PTK kernel each thread loads one state from the global memory to the register, generates its own sequence of random numbers during the PTK, and continually updates the associated state in the register. This effectively reduces overall memory latency, since accessing the on-chip space is substantially faster than the off-chip space. Secondly, the statically allocated space consists of a 64 KB constant memory and per-thread local memory. The constant memory is used to store constants such as Avogadro's number and electron rest mass that are frequently accessed by all the threads. On Fermi GPU, each thread is supplied with only 63 32-bit on-chip registers at most. If a kernel requests more than that amount to hold variables, the excess portion will be transferred to local memory which causes extra access latency (termed as register spill). To limit register usage, sometimes redundant calculations are made to replace the temporary variables stored in the registers. The third type of DRAM is the runtime heap which allows dynamic allocation of per-thread memory. Derivation of the attenuation coefficients from the cross-section data requires multiple intermediate arrays to be stored in this space.

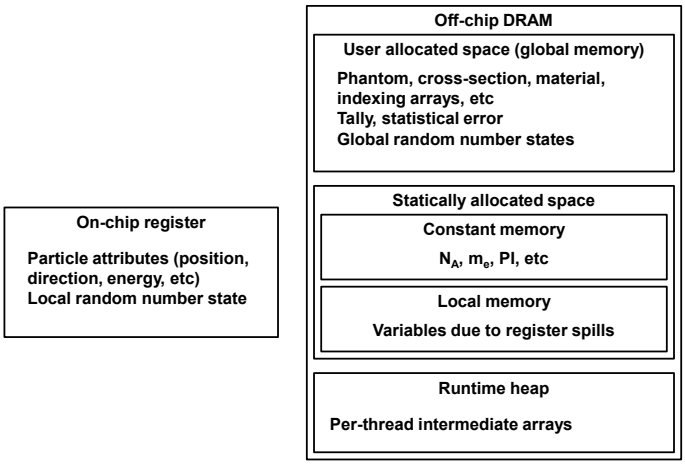

Figure 4. Memory layout of a GPU with Fermi-architecture.

The amount of energy deposited by the photon and the associated statistical error constitute the final dose result. Here we use one of two different methods to collect the result from all simulated photons, as is shown in Fig. 5. The first method, atomic summation, comes as a simple, CPU-like way. Once the energy of a photon is deposited into a certain organ due to photoelectric effect or incoherent scattering, the register that temporarily holds the deposited energy will immediately add the result to two locations: one is the user allocated space that records the accumulated dose value in that organ; the other is the per-thread runtime heap used to buffer and update the statistical error on a per-history basis. Here the "addition" operation to the user allocated space has to be ***atomic*** to avoid race conditions, where multiple threads compete in doing the read-and-modify operation on the same memory location simultaneously. Atomic addition serializes

the participating threads and guarantees all the values carried by them are safely added together. After the entire PTK ends the dose result in the user allocated space is copied to the host memory. The other method, parallel summation, aims at eliminating the serialization caused by atomic operations. In simulating each photon, the register immediately adds the deposited energy and the statistical error to the per-thread runtime heap. This process repeats until simulation of 100 photons assigned to the thread finishes. The dose result will then be copied to the user allocated space where each thread has its own place to hold the data. This is followed by an elaborately designed reduction algorithm developed by Nvidia to efficiently sum up the data in parallel and return the final result to the host. Note that this method has a trade-off: despite the avoidance of serialization by atomic operations, the increased use of runtime heap can introduce extra memory access latency.

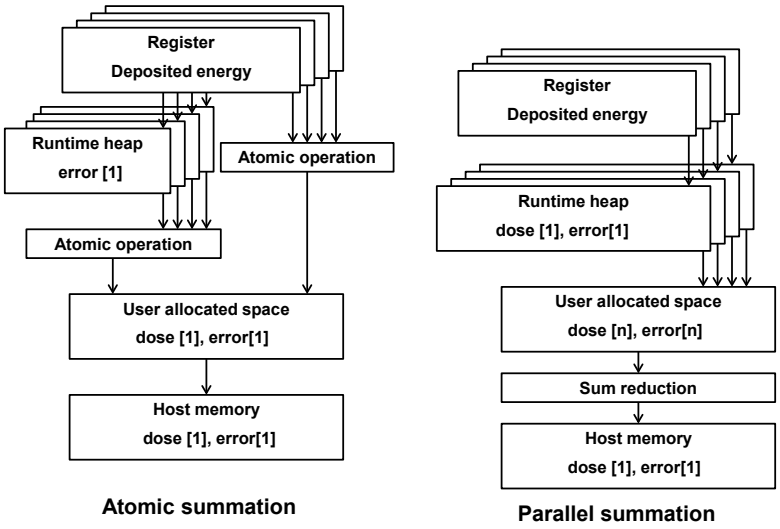

Figure 5. Diagram of two summation methods: atomic and parallel. This figure shows how the result of one organ dose is collected from n threads, which are represented by the partially overlapped boxes. The number in the square bracket refers to the size of the array.

2.5.3. Other issues

Single-precision floating point arithmetic is used in ARCHER considering the fact that its FLOPS is twice larger than that of double-precision arithmetic on Fermi-GPUs. Theoretically, the resultant representation error may affect the accuracy of every chain of the simulation. For our specific application, however, this error is tolerable except for atomic summation, as can be seen in the result section. To further improve the performance, the compiler option “fast math” is enabled for the device code, which reduces the accuracy of division operation and transcendental functions in the standard math library but as a payoff increases the speed by producing fewer assembly instructions.

3. RESULTS

3.1. Accuracy Benchmark

To evaluate the performance of $\mathrm{ARCHER_{GPU}}$, a whole-body scan is simulated. The patient model is a morbidly obese 142 kg RPI-Adult Male phantom with $135 \times 132 \times 509$ voxels. The scan protocols include 20 mm beam collimation, 120 kVp and axial mode with a pitch of 1 (90 axial scans are needed in total). For each axial scan, a total of 10^7 photons are simulated. The absorbed doses to 28 organs/tissues of dosimetric significance are tallied, covering all those listed in ICRP recommendations. Organs/tissues that constitute the "remainder" tissue defined in [19], are individually tallied. Parallel summation method is adopted to collect the dose result from different threads.

The organ doses obtained by $\mathrm{ARCHER_{GPU}}$ and MCNPX are listed in Fig. 6 in the form of signed percentage difference, defined as $(\mathrm{ARCHER_{GPU}} - \mathrm{MCNPX})/\mathrm{MCNPX} \times 100\%$. On average, the absolute percentage difference is 0.38%. The maximum discrepancy occurs to the thyroid dose which is -1.69% different from MCNPX. In our phantom the thyroid contains iodine with a weight fraction of 0.1%. This high-Z element tends to generate fluorescent x-rays with higher kinetic energy (33.2 keV for the 1s shell) in the aftermath of photoelectric effect. However, ignoring fluorescence transport is expected to produce an overestimate of the absorbed dose, which is the opposite of our test result. It is therefore inferred that the absence of Doppler broadening effect which flattens the energy spectrum of the incoherently scattered photon due to the pre-collision momentum of the electron is a possible reason and should be further investigated.

3.2. Performance Comparison

MCNPX was executed on a single CPU core using one thread. In comparison, $\mathrm{ARCHER_{CPU}}$ was run on the 6 cores using 6 MPI processes, and $\mathrm{ARCHER_{GPU}}$ on one GPU card with Error-Correcting Code (ECC) enabled. Although a total of 512 SPs are used, the actual number of GPU threads resident on a GPU at runtime is problem-dependent and in our test simulation is approximately 8,000 according to the CUDA program profiling result. The computation time is listed in Table I. While significantly faster than MCNPX, $\mathrm{ARCHER_{GPU}}$ has shown a speedup of 2.5× over $\mathrm{ARCHER_{CPU}}$ (6-core).

Table I. Comparison of the computation time. The CPU is an Intel Xeon 5650 6-core processor, and the GPU is an Nvidia Tesla M2090 card.

Simulation tool	Total number of particles simulated	Time [min]
MCNPX (1-core)	9×10^8	8,689
$\mathrm{ARCHER_{CPU}}$ (6-core)	9×10^8	139
$\mathrm{ARCHER_{GPU}}$ (1-card)	9×10^8	56

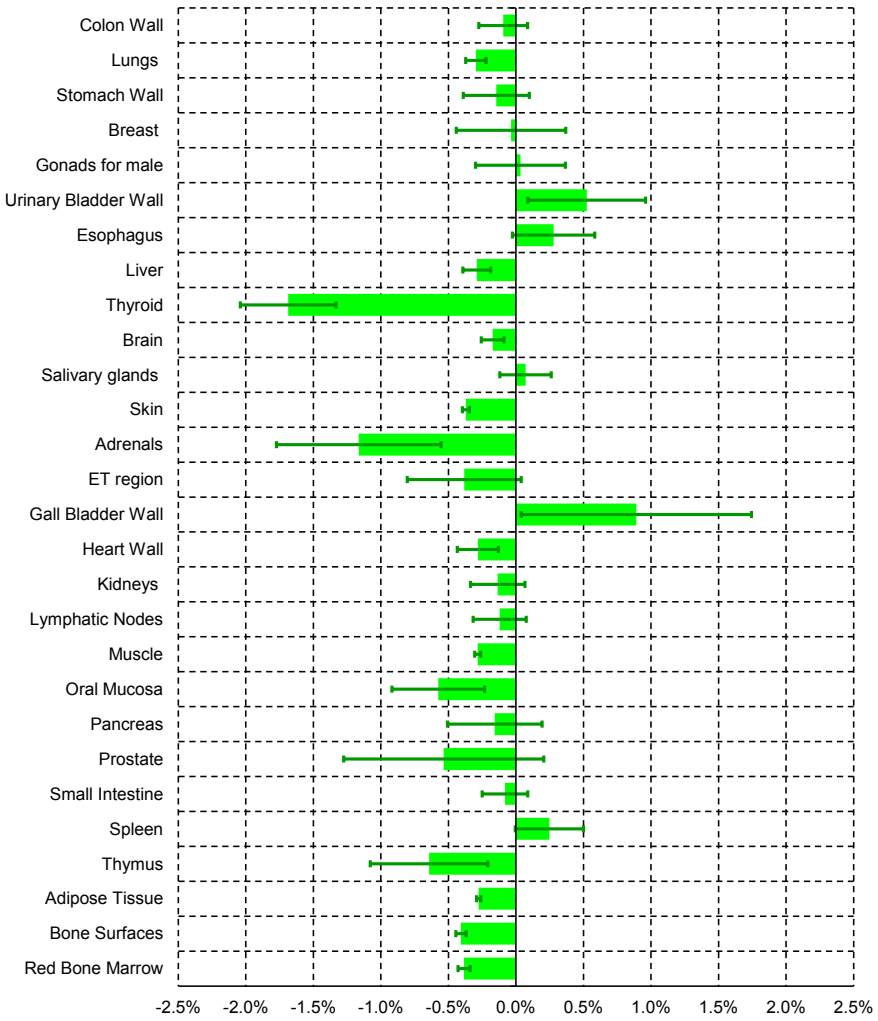

Figure 6. Percentage differences in dose results from $\mathbf{ARCHER_{GPU}}$ and MCNPX, calculated by $(\mathbf{ARCHER_{GPU}} - \mathbf{MCNPX})/\mathbf{MCNPX} \times 100\%$.

3.3. Comparison Between Atomic and Parallel Summation

Both atomic and parallel summation methods have their particular advantages over each other as mentioned in 2.5.2. For the former method, however, as the problem size scales up, the numerical error will become increasingly noticeable, as illustrated by Fig. 7. In this test case, dose to the lung in a chest region CT scan is calculated. The atomic summation fails after the number of photons exceeds 10^8, the result being smaller than the true value. This deviation is due to the fact that as the sum (energy deposition stored in user allocated memory space) becomes larger, more and more low-order digits of a small floating point number added to it are discarded. In contrast, the result obtained by parallel summation maintains good consistency, being 0.8% different from MCNPX result. In this method, which is based on the classic pairwise summation, the two floating pointer numbers added together are generally not several orders of magnitude different; hence a smaller numerical error.

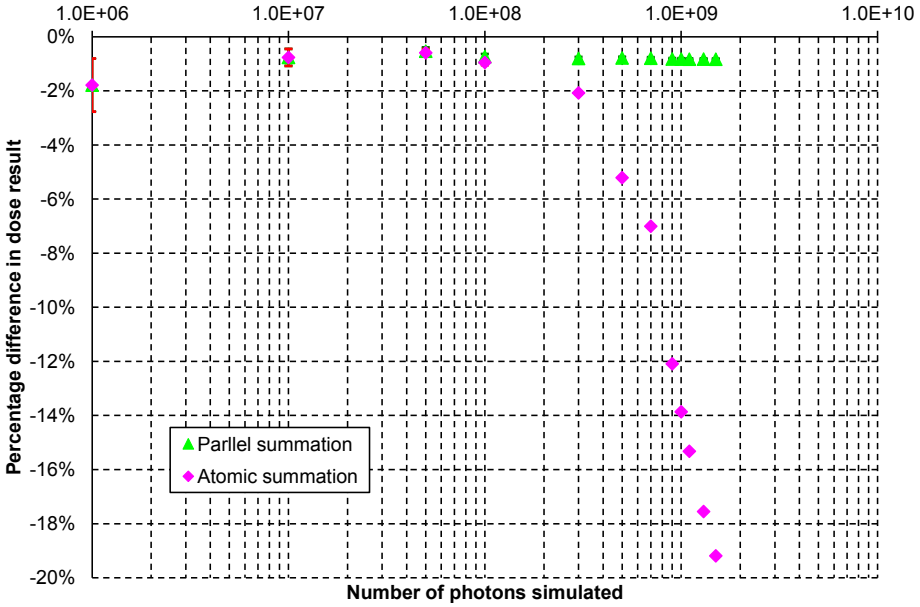

Figure 7. The influence of parallel and atomic summation methods over the accuracy of the lung dose in a simulated chest scan.

In theory, numerical errors due to the atomic operation can be removed by using the double precision floating point arithmetic. Currently, 64-bit floating point atomic addition is not directly supported by CUDA GPUs. Nvidia proposed a compare-and-swap algorithm that emulates the double precision arithmetic [26]. This emulation method, however, considerably reduces the overall computational performance and is not feasible in fast dose calculations.

4. DISCUSSIONS AND CONCLUSIONS

In this paper, we present our GPU-accelerated Monte Carlo simulation code $\mathrm{ARCHER_{GPU}}$ developed for fast CT imaging dose calculations. The new code effectively reduces the computation time of a whole-body CT scan simulation, while producing estimations of absorbed doses to organs/tissues with high accuracy. Because of this, we have used $\mathrm{ARCHER_{GPU}}$ as a substitute for the production code MCNPX in providing a massive amount of dosimetric data for VirtualDose, a web-based software under development to track and report patient CT organ doses [27].

As part of the NIH funded project, $\mathrm{ARCHER_{GPU}}$ will be integrated into a sub-minute Monte Carlo based CT dose evaluation and reporting system. Several efforts are being made to further improve the accuracy and performance of $\mathrm{ARCHER_{GPU}}$. First, the models of Doppler broadening caused by the pre-collision momentum of the electrons as well as the x-ray fluorescence generated from electron de-excitation will be added to eliminate the inaccuracy in the thyroid dose calculation. Second, a hybrid sampling strategy will be adopted to sample the photon's path-length. The current delta tracking method can be inefficient when photons are traveling in the surrounding air before they enter the phantom, because the small attenuation coefficients of air usually lead to frequent occurrence of virtual collisions and thus more path-length re-samplings, introducing extra computation time. To circumvent this problem, the traditional ray-tracing

method will be used to replace delta tracking in all the non-lattice geometries outside the phantom.

ACKNOWLEDGEMENTS

Grant support from National Institute of Biomedical Imaging and Bioengineering (R01EB015478).

REFERENCES

[1] C. McCollough et al., *The measurement, reporting and management of radiation dose in CT*, American Association of Physicists in Medicine, College Park, MD, USA (2008).

[2] M. K. Kalra et al., "Radiation exposure from chest CT: issues and strategies," *J Korean Med Sci.*, **19**, pp. 159-166 (2004).

[3] E. S. Amis et al., "American College of Radiology white paper on radiation dose in medicine," *J Am Coll Radiol.*, **4**, pp. 272-284 (2007).

[4] F. B. Brown, W. R. Martin, "Monte Carlo methods for radiation transport analysis on vector computers," *Progress in Nuclear Energy.*, **14**, pp. 269-299 (1984).

[5] S. W. Keckler et al., "GPUs and the Future of Parallel Computing," *Micro IEEE.*, **31**, pp. 7-17 (2011).

[6] "Top 500 supercomputer sites: statistics list," http://www.top500.org/statistics/list/ (2002).

[7] Jia et al., "Fast Monte Carlo simulation for patient-specific CT/CBCT imaging dose calculation," *Phys. Med. Biol.*, **57**, pp. 577-590 (2012).

[8] S. Hissoiny et al., "GPUMCD: A new GPU-oriented Monte Carlo dose calculation platform," *Med. Phys.*, **38**, pp. 754-764 (2011).

[9] W. Chen et al., "Fast on-site Monte Carlo tool for dose calculations in CT applications," *Med. Phys.*, **39**, pp. 2985-2996 (2012).

[10] A. Ding et al., "Extension of RPI-adult male and female computational phantoms to obese patients and a Monte Carlo study of the effect on CT imaging dose," *Phys. Med. Biol.*, **57**, pp. 2441-2459 (2012).

[11] Y. H. Na et al., "Deformable adult human phantoms for radiation protection dosimetry: anthropometric data representing size distributions of adult worker populations and software algorithms," *Phys. Med. Biol.*, **55**, pp. 3789-3811 (2010).

[12] J. Gu et al., "The development, validation and application of a multi-detector CT (MDCT) scanner model for assessing organ doses to the pregnant patient and the fetus using Monte Carlo simulations," *Phys. Med. Biol.*, **54**, pp. 2699-2717 (2009).

[13] J. Zhang et al., "A pair of mesh-based, size-adjustable adult male and female computational phantoms using ICRP-89 parameters and their calculations for organ doses from monoenergetic photon beams," *Phys. Med. Biol.*, **54**, pp. 5885-5908 (2009).

[14] X-5 Monte Carlo Team, *MCNP-A general Monte Carlo N-Particle transport code version 5*, Los Alamos National Laboratory, USA (2003).

[15] E. D. Cashwell et al., *Monte Carlo Photon Codes: MCG and MCP*, Los Alamos National Laboratory, USA (2003).

[16] R. N. Blomquist, E. M. Gelbard, "An Assessment of Existing Klein-Nishina Monte Carlo Sampling Methods," *Nucl. Sci. Eng.*, **83**, pp. 380-384 (1983).

[17] L. Jaakko, "Performance of Woodcock delta-tracking in lattice physics applications using the Serpent Monte Carlo reactor physics burnup calculation code," *Annals of Nuclear Energy*, **37**, pp. 715-722 (2010).

[18] "The stopping power and CSDA range of electrons in material," http://physics.nist.gov/PhysRefData/Star/Text/ESTAR.html (2002).

[19] ICRP, "ICRP Publication 103: the 2007 Recommendations of the International Commission on Radiological Protection," *Annals of the ICRP*, (2007).

[20] H. Schlattl, "Organ dose conversion coefficients for voxel models of the reference male and female from idealized photon exposures," *Phys. Med. Biol.*, **52**, pp. 2123-2145 (2007).

[21] V. R. Nowotny, "Ein Programm für die Berechnung von diagnostischen Röntgenspektren," *RöFo.*, **142**, pp. 685-689 (1985).

[22] Nvidia, *Whitepaper of Nvidia's next generation CUDA compute architecture: Fermi*, Nvidia (2009).

[23] "Nvidia CURAND Library," http://docs.nvidia.com/cuda/curand/index.html (2002).

[24] A. Ding et al., "Evaluation of speedup of Monte Carlo calculations of simple reactor physics problems coded for the GPU/CUDA environment," *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)*, Rio de Janeiro, RJ, Brazil, May 8-12, 2011, (2011).

[25] T. Liu et al., "A Monte Carlo neutron transport code for eigenvalue calculations on a dual-GPU system and CUDA environment," *PHYSOR 2012 Advances in Reactor Physics Linking Research, Industry, and Education*, Knoxville, TN, USA, April 15-20, 2012, (2012).

[26] Nvidia, *CUDA C programming guide v5.0*, Nvidia (2012).

[27] A. Ding et al., "Design and testing of the VirtualDose software under the Software as a Service (SaaS) platform for tracking and reporting CT doses," *Med. Phys.*, Charlotte, NC, USA, July 29-August 2, Vol. 39, pp. 3876 (2012).