DEVELOPMENT OF ARCHER — A PARALLEL MONTE CARLO RADIATION TRANSPORT CODE — FOR X-RAY CT DOSE CALCULATIONS USING GPU AND COPROCESSOR TECHNOLOGIES

By

Tianyu Liu

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: NUCLEAR ENGINEERING AND SCIENCE

Approved by the
Examining Committee:

_____
Xie George Xu, Thesis Adviser

_____
Peter F. Caracappa, Member

_____
Christopher D. Carothers, Member

_____
Yaron Danon, Member

_____
Wei Ji, Member

Rensselaer Polytechnic Institute
Troy, New York

July 2014
(For Graduation August 2014)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I have had the great honor to be part of Rensselaer Radiation Measurement & Dosimetry Group (RRMDG) at Rensselaer Polytechnic Institute (RPI), and work with a team of the most diligent and intelligent people I have ever met. To me these 5-year research experience is an invaluable treasure to be cherished in my life.

I would express my deep sense of gratitude to Professor X. George Xu, the leader of RRMDG and my research advisor. He has a pioneer spirit with considerable foresight and courage, and always leads me to new, challenging areas that few have trodden before. I especially appreciate the abundant research resources he has provided to me, and a great deal of advices to help me think deeply and broadly and act wisely and efficiently.

I am profoundly indebted to my doctoral committee members, including Professor Christopher D. Carothers, Professor Peter F. Caracappa, Professor Wei Ji and Professor Yaron Danon, for their guidance and encouragement. Special thanks are due to Professor Mark S. Shephard for his extremely insightful advices on this research. Many thanks are given to Professor Forrest B. Brown from Los Alamos National Laboratory for the enormous help with algorithm design. He is extraordinarily nice and outstandingly knowledgeable, and is the kind of scientist I want to be.

I would like to thank the physicists, clinicians and researchers from Massachusetts General Hospital, including Dr. Bob Liu, M.D. Mannudeep K. Kalra, Dr. Jim Q. Shi, Dr. Da Zhang, Dr. Xinhua Li, Dr. Wenli Cai for the significant amount of efforts and time they have contributed to this project.

Besides, I would like to thank my colleagues and friends (alphabetical order): Dr. Aiping Ding, Dr. Xining Du, Mr. Yiming Gao, Mr. Deyang Gu, Dr. Jianwei Gu, Dr. Bin Han, Dr. Hua Li, Dr. Yanheng Li, Dr. Chao Liang, Miss Wenyan Liu, Dr. Matthew Mille, Dr. Yong Hum Na, Miss Elise Noel, Miss Theresa B. Tram Phamduy, Mr. Matthew Riblett, Dr. Lin Su, Mr. Justin Vazquez and Mr. Noah Wolfe. They have not only helped me tremendously with my research but also with

my life.

I am also particularly grateful to those who have offered altruistic, unreserved help to me. They include Dr. Bart Willems from Atipa Technologies who provided valuable instructions on Linux administration, the anonymous expert with username "talonmies" on Stackoverflow who has enlightened me as to GPU programming with his profound knowledge.

Much appreciation is owed to Health Physics Society (HPS) which granted me 2010-2011 Richard J. Burk, Jr. Fellowship, to Nvidia Corporation which made generous hardware donation to our research group, and to the National Institute of Biomedical Imaging and Bioengineering (NIBIB) which provided financial support to this project (R01EB015478).

My research would not be possible without the constant, unswerving support from my family. I owe a debt of gratitude to my parents Bing Liu and Xiuping Na, and my fiancée Ning Chen. Ning's Mike Wazowski's style of optimism is contagious and has helped me overcome many difficulties in my research.

ABSTRACT

Monte Carlo methods are the gold standard in radiation dose calculations with heterogeneous patient geometries and complicated irradiation conditions such as multi-detector CT scan. The long computation time has historically prevented them from becoming a routine clinic tool. The emerging hardware accelerators have created opportunities to speed up the computations significantly. They have the advantages of high computing power and high energy efficiency that are particularly suited for performing parallel tasks. This research represents our efforts to understand and utilize such technology in the context of radiation dosimetry, and is focused on developing and testing a new parallel Monte Carlo package, named ARCHER, for patient-specific CT dose calculations using three types of hardware platforms, including the conventional multi-core CPU and two most competitive hardware accelerators — the Nvidia's graphics processing unit (GPU) and Intel's Xeon Phi coprocessor. ARCHER includes three variants, $\text{ARCHER}_{\text{CPU}}$, $\text{ARCHER}_{\text{GPU}}$ and $\text{ARCHER}_{\text{COP}}$, which are tested on a 6-core Intel Xeon X5650 CPU, three Nvidia GPUs (M2090, K20, K40), and an Intel Xeon Phi 5110p coprocessor, respectively. ARCHER has a built-in model of the GE LightSpeed Pro 16 CT scanner and a library of computational human phantoms that allow realistic scan protocols to be simulated. For a fair code comparison, all the variants are carefully optimized and fine-tuned to their specific hardware platforms. Important performance factors such as the accuracy, computing efficiency, scalability and energy efficiency of the codes are investigated. The accuracy tests include the benchmark of the Monte Carlo transport kernels against the production Monte Carlo code MCNPX, and the benchmark of the simulation models against the experiment using a real human subject. In the first test, ARCHER is in excellent agreement with MCNPX using the same geometries and similar physics. In the second test, discrepancy up to 29% from the experiment is observed, encouraging us to reinvestigate the simulation models such as the CT scanner model and the algorithm to automatically generate the phantom from CT images. In the computing efficiency test, compared to the parallel CPU code,

$\text{ARCHER}_{\text{GPU}}$ is found to be faster by a factor of $5.40 \sim 10.89$, while $\text{ARCHER}_{\text{COP}}$ is by a factor of 3.37. $\text{ARCHER}_{\text{GPU}}$ demonstrates good scalability when the GPU stream is implemented. The GPU platform is found to be the most energy-efficient, consuming less amount of energy than the CPU by a factor of $3.68 \sim 8.01$, while the coprocessor is better than the CPU by a factor of 2.24. Meanwhile, both the GPU and coprocessor platforms are found to be more cost effective than the CPU. Furthermore, $\text{ARCHER}_{\text{GPU}}$ is applied to a clinical case to compute imaging dose distributions in a patient-specific abdominal CT scan and exhibits good computing efficiency. This research shows that both the GPU and the coprocessor technology can effectively boost the performance of Monte Carlo simulations, that the GPU takes the clear lead, and that the developed code ARCHER is an important step toward patient-specific CT dose calculations.

CHAPTER 1
INTRODUCTION

"By an incredible coincidence, Gamow and Edward Condon, who had discovered simultaneously and independently the explanation of radioactivity (one in Russia, the other in this country), came to spend the the last ten years of their lives within a hundred yards of each other in Boulder."

—Ulam, Stanislaw

1.1 Background

Clinical use of X-ray Computed Tomography (CT) — an important diagnostic imaging tool — has continued to grow on a yearly basis. This trend has led to a significant increase in the radiation dose delivered to the patient population, giving rise to a mounting concern over the public health. The radiology community has called for the development of new tools to more accurately quantify and report *patient-specific* dose for CT scan procedures. The Monte Carlo methods are one of the ideal candidates, being in general deemed as the gold standard in radiation dose calculations due to its high accuracy. It allows exact modelling of three-dimensional, heterogeneous geometries, includes precise mathematical models for radiation particle interactions with matter, and adopts a cross-section representation of the physics

Portions of this chapter previously appeared as: Liu, T., Ji, W. & Xu, X. G. (2013), Development of GPU-based Monte Carlo code for fast CT imaging dose calculation on CUDA Fermi architecture, *in* 'International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)', Sun Valley, ID, pp. 1199-1210.

Portions of this chapter are to appear in: Liu, T., Du, X., Su, L., Ji, W., Carothers, C. D., Shephard, M. S., Liu, B., Kalra, M., Brown, F. B., Fitzgerald, P. F. and Xu, X. G. (2014), 'ARCHER-CT, an extremely fast Monte Carlo code for patient-specifc ct dose calculations using NVIDA GPU and Intel coprocessor technologies: part I — software development and testing', *Phys. Med. Biol.*. (submitted).

that is as accurate as experiments permit (Brown & Martin 1984). Nonetheless, using Monte Carlo methods, the computation time to obtain results with acceptable statistical uncertainties is usually very long, making a routine use in a clinical setting impractical. The computation can be effectively accelerated by developing parallel Monte Carlo codes. Traditional parallel codes execute on the systems composed of central processing units (CPU). Recent advance in High Performance Computing (HPC) industry has introduced new paradigms for parallel computing — the hardware accelerators such as the graphics processing unit (GPU) and the coprocessor. They have the distinctive advantage of high computing power and high energy efficiency. Existing production codes, however, cannot be directly run on them. To harness this emerging technology for accurate and fast CT dose calculations, it is necessary to redevelop the Monte Carlo code and optimize it to the unique GPU or coprocessor architecture (Liu et al. 2013). This is the central task of this doctoral research.

1.2 Clinical Significance of CT Dose Management

Since its inception some 40 years ago, computed tomography (CT) has become one of the most widely used medical imaging tools. The number of CT scans performed each year has been steadily increasing by a factor of 10% to 15% (Amis Jr et al. 2007, Brenner & Hall 2007, NCRP 2009, McCollough et al. 2008). CT scans now account for about one in five radiation-based imaging procedures and these scans are responsible for nearly 50% of the radiation exposure from medical imaging in the U.S. (NCRP 2009, Stern 2007, Stern et al. 2000). Over the years, CT has played a significant role in cancer treatment with three-dimensional anatomical data routinely incorporated into treatment planning for radiation oncology. From 1995 to 2007, the use of CT scans in emergency department visits has increased by approximately 6-fold in the U.S. (Larson et al. 2011). Associated with this popularity is the radiation dose level to the population — the U.S. annual per capita dose by CT scans has increased from 0.03 mSv to 1.47 mSv during that time (NCRP 2009).

For years, controversy about the potential risk of induced carcinogenic effects has surrounded clinical practices involving X-ray radiography and CT as is men-

tioned in Brenner (2002), Brenner & Hall (2007) and National Research Council (2005, pp. 65–90). The risk to an individual patient of developing a radiation-related cancer from any single CT procedure is estimated to be relatively small according to data derived from nuclear workers and atomic bomb survivors (Brenner & Hall 2007, NCRP 2009). However, there is uncertainty in applying these high-dose and high-dose rate radiobiological data to medical radiation exposures such as CT. Recent epidemiological studies on CT patients have provided more relevant information. One study reported by Berrington de González & Darby (2004) hypothesized that medical exposures between 1991 and 1996 might be responsible for approximately 1% of all cancer incidences in the United States during that period. The issue of radiation risk is of even greater importance for pregnant and pediatric patients, since younger patients are known to be considerably more radiosensitive. A study by Pearce et al. (2012) found that a dose of 50 mGy from CT tripled the risk of leukaemia and a dose of 60 mGy tripled the risk of brain cancer among the 178,604 young patients who received CT scans from 1985 $\sim$ 2002 from 81 hospitals in Great Britain. The increased risk to young patients was also observed in a study by Mathews et al. (2013) that analyzed 10.9 million Australian patients who were between 0 to 19 years old.

The "As Low As Reasonably Achievable" (ALARA) principle is widely adopted in industrial and medical radiation protection as a prudent measure. To address the increasing trend in CT exposure, professional societies have implemented a number of aggressive initiatives (Amis Jr et al. 2007, Goske et al. 2008, McCollough et al. 2008, ICRP 2007, NCRP 2009). In 2011, a workshop cosponsored by the National Institute of Biomedical Imaging and Bioengineering (NIBIB) was held to identify possible research steps towards reducing the average patient CT exposure to sub-mSv levels and, subsequently, a special U01 research program was launched by NIBIB in 2012 (NIH. 2012).

Current CT scanners only report CT Dose Index (CTDI) values and dose length product (DLP) that are based on data pre-measured in tissue-equivalent cylinders (McCollough et al. 2008). Recent studies have concluded that CTDIs and DLPs are poor surrogates for patient CT doses (Li et al. 2010, 2011). In an attempt

to move away from the CTDI concept toward patient-relevant CT dose reporting, AAPM recently designed Size-Specific Dose Estimates (SSDE) (Boone et al. 2011).

The use of the unit "mSv" in the NIBIB's research initiative implies the use of the "effective dose" that was originally defined by the ICRP for estimating whole-body stochastic risk to workers. In the ICRP radiation protection dose system, estimates of radiation levels are based on the absorbed dose to radiosensitive organs (instead of "dose at a point" in the body) for which radiation risk information exists (Health Physics Society. 2010, ICRP 2007). Correct uses of "effective dose" for CT imaging have been discussed (see for examples, McCollough & Schueler (2000), McNitt-Gray (2002), Brenner & Huda (2008), Xu et al. (2008)). The capability of patient-specific organ dose assessment for X-ray based imaging procedures has been proposed before (Li et al. 2010, 2011) and is clearly a trend in radiology research.

Several software tools are available for CT organ dose and effective dose estimate, such as ImPACT (Lewis 2005) and VirtualDose (Ding et al. 2012*b*, Virtual Phantoms Inc. 2013). In principle, they can retrospectively re-construct organ doses based on pre-calculated database using the population-averaged computational phantoms (Xu & Eckerma 2010, pp. 347–377). VirtualDose, for example, utilizes an extensive organ dose database derived from time-consuming Monte Carlo calculations for a library of patient phantoms representing pregnant and obese adult patients as well as age-specific children (Virtual Phantoms Inc. 2013). These tools cannot perform accurate, patient-specific dose calculations, and will be well complemented by a code with the capability of efficient onsite Monte Carlo simulations (Liu et al. 2014*a*), which is addressed in this research.

1.3 Monte Carlo Methods

In radiation transport, the Monte Carlo methods are a computational algorithm that uses repeated random sampling to create the full genealogical histories of particles, and obtains numerical estimate of certain quantities, such as flux or fluence, that follow usually unknown probability distributions. The use of random sampling dates back to 18th century (X-5 Monte Carlo Team 2003*a*). Buffon (1777) posed and solved the Buffon's needle problem, which asks the probability of a nee-

dle landing on a line, given a floor marked with equidistant parallel lines. The result can be used to experimentally estimate π. The problem was later extended by Buffon and Laplace to the Buffon-Laplace needle problem (Arnow 1994), where the texture of the floor was changed from parallel lines to grids, and the probability of a needle landing on any one line of the grid was sought. In early 1900s Kelvin (1901) applied the random sampling to integral calculations in thermodynamics. In 1930s Fermi (X-5 Monte Carlo Team 2003*a*) invented a form of Monte Carlo method for neutron moderation study. During World War II, a team of eminent scientists, including Fermi, Ulam, von Neumann, Metropolis were working at Los Alamos on the Manhattan Project to develop the first atomic bomb (X-5 Monte Carlo Team 2003*a*). Their joint effort for neutronic computation continued after the war and was significantly encouraged by the advent of the first electronic computer Electronic Numerical Integrator And Computer (ENIAC). The computationally expensive random sampling procedure previously deemed impractical was now able to be put into practice, and was formally renamed as Monte Carlo method (X-5 Monte Carlo Team 2003*a*, Metropolis 1987). In 1947, von Neumann devised the first Monte Carlo program to solve neutron diffusion and multiplication problems on the ENIAC (X-5 Monte Carlo Team 2003*a*). Since then, the Monte Carlo method and radiation transport code have rapidly evolved. From 1963 to today, a general-purpose, continuous-energy, generalized-geometry, time-dependent, coupled neutron/photon/electron code called Monte Carlo N-Particle (MCNP) have been developed and upgraded by Los Alamos National Laboratory (X-5 Monte Carlo Team 2003*a*). Two renowned versions MCNP5 and MCNPX (Pelowitz 2008) have recently been merged into MCNP6 (Pelowitz 2013*b*). The code has nowadays become an international standard for a wide spectrum of applications (Selcow & McKinney 2000), including radiation protection and dosimetry (Ding et al. 2012*b*), radiation shielding (El-Guebaly 1997), radiography (Kardjilov et al. 2005), medical physics (Rogers 2006), nuclear criticality safety (Şeker & Çolak 2003), detector design and analysis (Childress & Miller 2002), nuclear oil well logging (Serov et al. 1998), accelerator target design (Overberg et al. 1999), fission and fusion reactor design (Wu 2006), decontamination and decommissioning (Love et al. 1995), nuclear waste storage and

disposal (Bayoumi et al. 2012), etc. Several other Monte Carlo codes also have large user base among medical physics community. They are Geant4 (Agostinelli et al. 2003), EGSnrc (Kawrakow & Rogers 2000), Penelope (Baró et al. 1995) and Fluka (Ferrari et al. 2005).

The Boltzmann transport equation can be solved by the Monte Carlo methods and the deterministic methods (such as discrete ordinates, integral transport, finite difference and finite element methods (Brown & Martin 1984)). The deterministic methods have an inherent disadvantage that discretization of the "time-space-angle-energy" phase space introduces approximations and computational systematic errors. Besides, the problem geometry and the level of detail to describe the interactions may be subject to a priori restrictions (Brown & Martin 1984). In contrast, the Monte Carlo methods do not have such disadvantages. According to Brown & Martin (1984) and Lewis & Miller (1984, pp. 296–356), the Monte Carlo methods adopt highly accurate mathematical models for particle interactions with matter, apply continuous treatment of phase space that obviates discretization errors, and permit exact modelling of three-dimensional, heterogeneous geometries. In this regard, the Monte Carlo methods are considered the most general and powerful numerical method available for solving radiation transport problems (Brown & Martin 1984).

The inherent statistical error of the result by Monte Carlo calculations can be controlled to an arbitrarily low level by simulating sufficiently many particles. In CT dosimetry study, for instance, it is a common practice to reduce the statistical uncertainty to 1% and below, which is often more precise than the error of experimentally obtained results. However, the computation time required to achieve this level of precision can be very long. This constitutes the *only* drawback of Monte Carlo methods (Brown & Martin 1984) and hinders them from being applied to routine calculations beyond a benchmark tool alone.

The Monte Carlo computer programs in general are *embarrassingly parallel*, meaning that the simulation task for individual particle history can often be carried out independently of one another with very small amount of communication between the tasks. Because of this attribute, the time-consuming sequential run can naturally benefit from parallel computing techniques. In 1980s, the "vectorized" MC codes

were elaborately designed by Brown & Martin (1984) and Bobrowicz et al. (1984) to run specifically on the vector computers of the time. In 1990s, the Parallel Virtual Machine (PVM) (Geist et al. 1994, pp. 93–135) and Message Passing Interface (MPI) (Gropp 2002) enabled parallelism across dozens of processors. Since 2000s, the invention of multi-core processors has allowed multiple threads to run concurrently on different cores of the same processor. To take advantage of the large-scale High Performance Computing (HPC) systems, many of the existing production codes have been reworked using a combination of the threading and MPI paradigms (Brown 2011). In addition, use of the network-based distributed computing system such as the clouding computing has also been reported lately (Wang et al. 2011, Miras et al. 2013).

1.4 New Parallel Computing Paradigm

In recent years, technical advances in the hardware architecture has introduced an alternative approach — to implement the Monte Carlo calculations on the *heterogeneous computing systems*. These systems feature the hardware accelerators attached to the conventional central processing units (CPU) (Brown 2011). Examples are the graphics processing unit (GPU), coprocessor, Field-Programmable Gate Array (FPGA), Cell processor, etc. They are playing an increasingly important role in HPC community and have been adopted in a growing number of supercomputer systems worldwide according to the statistics from the *Top 500 list* (Top500. 2013), illustrated in figure 1.1. By November 2013, 53 out of the top 500 supercomputer have been boosted by them; the No. 1 supercomputer Tianhe-2 (also known as MilkyWay-2) hosted by China National Super Computer Center has used 48,000 Intel Xeon Phi 31S1P coprocessors, while the No. 2 supercomputer Titan hosted by Department of Energy (DOE), DOE Office of Science (SC) and Oak Ridge National Laboratory has used 18,688 NVIDIA K20x GPUs (Top500. 2013). The academic publications on hardware accelerator-based Monte Carlo calculations is also on an uptrend.

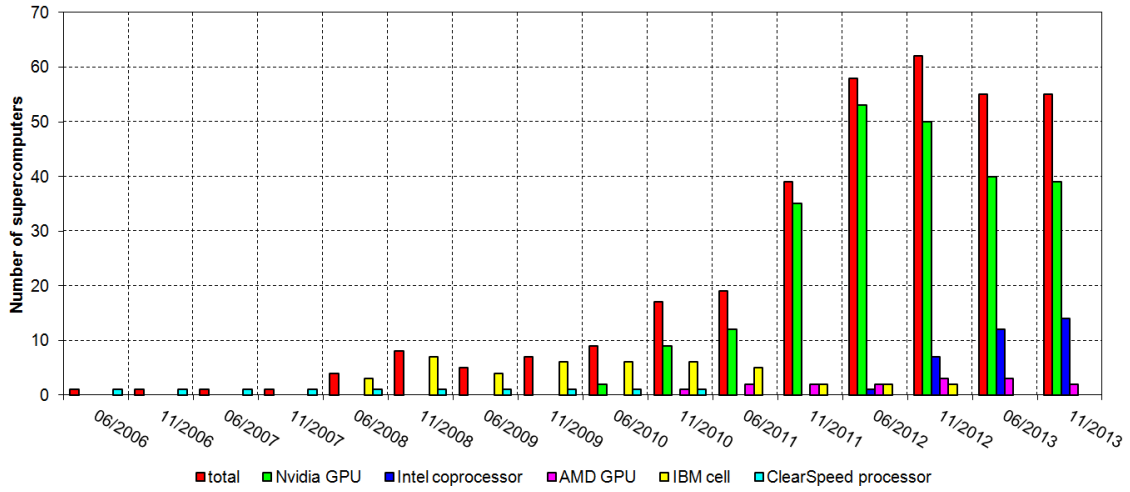


Figure 1.1: Statistics of Top 500 supercomputers worldwide (Top500. 2013) using hardware accelerators. Since 2006 the hardware accelerators have been utilized by a growing number of systems. The Nvidia GPU and Intel Xeon Phi coprocessor stand out as the two most used ones.

1.4.1 Advantage of Hardware Accelerators

The Nvidia GPU and Intel Xeon Phi coprocessor are distinguished from other accelerators by their rapid development and expanding domain of applications. One of their main advantages is the high energy efficiency, i.e. the ratio of the delivered computing performance to the consumed electric power. In the processor manufacturing industry, there is a heuristic called Moore's law (Moore 1998), which states that the number of transistors on integrated circuits doubles approximately every two years. It projects that we will arrive at the exascale computing era by around 2020, by which time the No. 1 supercomputer can have a performance of at least 1 exaFLOPS (10^{18} FLOPS). In developing this system, there will be four unprecedented challenges identified by Kogge et al. (2008), including energy and power, memory and storage, concurrency and locality, and resiliency, and the energy and power problem is given the top priority. Because of that, the energy-efficient GPU and coprocessor are considered a special candidate to enable the future supercomputer systems. In fact, by November 2013, the first 10 most energy-efficient supercomputers worldwide on the *Green 500 list* (Green500. 2013) have all used Nvidia GPUs, and No. 37 $\sim$ No. 42 supercomputers have all used Intel Xeon Phi

coprocessors.

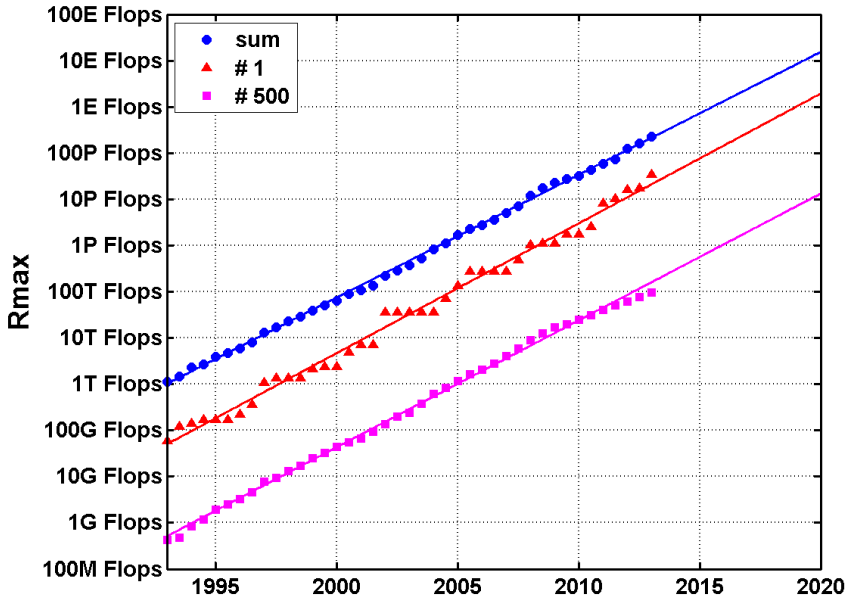


Figure 1.2: Performance of Top 500 supercomputers over the years (Top500. 2013). R_{max} refers to the maximal LINPACK performance achieved in unit of FLoating-point Operations Per Second (FLOPS). The increase in the performance of No. 1, No. 500 and the total performance nowadays continues to follow Moore's Law.

Another main advantage of the hardware accelerators is the high computing power — large FLOPS to be specifically — they are able to deliver on a single device. The Moore's law has been proven valid for approximately 50 years, as can partly be reflected by the trend of the Top 500 supercomputers' maximum performance over time in figure 1.2. Due to technical and economic barriers, however, this law has been challenged and will unlikely hold true indefinitely (Mack 2011). None the less, the GPU and coprocessor could add a significant amount of computing power to make the overall system performance keep pace with the projection of Moore's law, and thus arguably extend its validity to a longer period of time, as is suggested by Dally (2010) and Jeffers & Reinders (2013, pp. 1–22). For example, the LINPACK benchmark tests (Phillips & Fatica 2010, Intel. 2013*f*) performed by Nvidia and Intel demonstrated that on a single node with 2 CPUs and 2 hardware accelerators

as the additional computing units, the achieved maximal performance of the whole heterogeneous system was 5.6 $\sim$ 8.2 times higher than that of 2 CPUs alone.

1.4.2 GPU Architecture and Programming Model

The graphics processing unit (GPU) was originally invented by Nvidia in 1999 (Nvidia. 2011). At that time it was a fixed function graphics pipeline used typically for gaming purposes to process two types of graphics-specific programs called vertex shader and pixel shader (Lindholm et al. 2008). Since then a rapid development of the GPU technology has taken place. In around 2006, Nvidia introduced a parallel computing platform called Compute Unified Device Architecture (CUDA) that encompassed significant innovations on the compute architecture (hardware) and programmability (software) of the GPU (Nvidia. 2011). CUDA enabled the GPU to become more general-purpose such that it can solve wider range of scientific parallel problems beyond computer graphics alone. It also allowed the programmers to write code in the common, high-level programming languages such as C/C++ and Fortran instead of the specialized graphics application programming interfaces (API) (Nvidia. 2011). So far the CUDA GPU has had several generations, including Tesla, Fermi, Kepler, Maxwell and Pascal, each being accentuated by an additional technical innovation. For example, the first Tesla generation (released in $\sim$ 2008) introduced CUDA itself; the past Fermi generation (released in $\sim$ 2010) supported *64-bit floating point operations*, expanding the range of GPU applications; the current Kepler generation (released in $\sim$ 2013) introduces *dynamic parallelism*, reducing the communication cost between CPU and GPU; the upcoming Maxwell generation (expected to be released in $\sim$ late 2014) will provide compatibility with Microsoft's new multimedia API collection DirectX 12; the future Pascal generation (expected to be released in $\sim$ 2016) will feature *unified virtual memory* (allowing the system memory and GPU memory to be addressed in a single space) and *3-D stacked memory* (permitting more memory per unit of volume), substantially increasing the memory size, bandwidth and the ease of programmability (Purches 2013).

Nvidia's CUDA GPU adopts a hierarchical design in terms of the programming abstraction, processing hardware and memory hardware (Hennessy & Patterson

2012, pp. 288–315), explained as follows.

- *Programming abstraction* A CUDA code has two components: the *host code* that is essentially the same with the conventional CPU code and executes in sequence on the CPU, and the *device code* that executes in parallel on the GPU. The device code consists of one or more *kernels*, which simply refers to the user-developed GPU-specific functions. A kernel is executed by a large number of parallel GPU *threads* (Nvidia. 2011). A certain number of threads form a higher level of unit called *block*, and a certain number of blocks form the highest level of unit called *grid*. The thread, block and grid constitute CUDA's programming hierarchy from the bottom to the top. Users can specify the number of threads per block and the number of blocks per grid launched on the GPU. Besides, there is an additional concept called *warps*. A warp is a bundle of 32 threads, and is an implementation of Nvidia's Single-Instruction, Multiple-Thread (SIMT) model wherein the 32 threads always execute a common instruction (Nvidia. 2013*b*).

- *Processing hardware* A CUDA GPU has a large number of basic processing units called *Streaming Processor* (SP), also known as CUDA core. A number of SPs along with additional hardware accessories form a higher level of unit called *Streaming Multiprocessor* (SM). A number of SMs with additional accessories form the entire GPU (Nvidia. 2011). There is a close relation between the processing hardware and the programming abstraction (Seibert. 2011). When a kernel is launched, a GPU global thread scheduler automatically distributes the blocks to the available SMs for execution (Nvidia. 2011). A block can reside in only one SM, and an SM can hold one or more blocks if hardware resource permits. On each SM, a local warp scheduler automatically organizes the execution of warps within the resident blocks. The instructions from a warp are issued to a group of SPs on the SM. Different warps from the same or different blocks may be interleaved and executed concurrently. It is worth pointing out that on a CPU, one can explicitly attach threads or processes to cores, but on a GPU, the mapping between blocks or warps to SMs

is entirely automated by the hardware itself and currently does not permit users' intervention.

- *Memory hardware* A CUDA GPU has on-chip memory located on each SM, and off-chip memory outside of them (Nvidia. 2013*b*). From a programming perspective, these memories can be divided into several categories corresponding to the hierarchical programming abstraction. A thread has an exclusive space on the on-chip memory called *register* to store local variables, and also a space on the off-chip memory called *local memory* to store excess variables that the register cannot hold. A block has an exclusive space on the on-chip memory called *shared memory* shared between threads within that block. A grid has a *global memory* and *constant memory* space on the off-chip memory, both shared among all the threads across the whole grid. Table 1.1 summarizes the property of these memories.

Table 1.1: Memory of Kepler GPU. The table is a modification to the one from Nvidia. (2013*a*). The scope refers to the level of programming abstraction that can access the memory. The lifetime refers to the duration of that accessibility.

memory	register	local	shared	global	constant
managed by	compiler	compiler	programmer	programmer	programmer
location	on-chip	off-chip	on-chip	off-chip	off-chip
cached	no	yes	no	yes	yes
access	read and write	read and write	read and write	read and write	read only
scope	per thread	per thread	all threads per block	all threads per grid	all threads per grid
lifetime	thread	thread	block	host allocation	host allocation

Apart from CUDA, the Nvidia GPU also supports several other types of programming models, summarized in Table 1.2.

Table 1.2: Programming models supported by the GPU and coprocessor. The coprocessor allows more programming models.

hardware accelerator	programming model supported
GPU	CUDA, OpenCL, OpenACC, OpenMP
coprocessor	MPI-OpenMP, MPI-Pthreads, Cilk, TBB, offload pragma, OpenACC, OpenMP

1.4.3 Coprocessor Architecture and Programming Model

In 2008, Intel introduced their own graphics pipeline architecture codenamed "Larrabee". It bundled many in-order CPU cores running an extended version of the x86 instruction set (Seiler et al. 2008). In 2010, based on the Larrabee processor, Intel started to develop their new, many-core coprocessor codenamed "Many Integrated Core" (MIC) for general-purpose computing (Intel. 2010). The initial prototype was named "Knights Ferry," with 45 nm process, 32 CPU cores operating at 1.2 GHz, and 2 GB on-board memory tantamount to GPU's off-chip memory (Intel. 2010). In around 2012, several major improvements were made to the coprocessor, including enhanced semiconductor manufacturing processes (22 nm), more number of CPU cores (more than 50) and larger onboard memory (6 GB) (Intel. 2012). The new coprocessor was named "Knights Corner," later rebranded as "Intel Xeon Phi" and became publicly available. The next generation of Intel's many-core product will be named "Knights Landing," featuring the 14 nm process and large on-package memory tantamount to GPU's on-chip memory. The product will appear either as the standalone CPU or as the hardware accelerator (Wechser 2014).

The coprocessor does not require a programmer to have as much intimate knowledge on the hardware architecture as the GPU does. In practice, it can be directly viewed and used as a many-core CPU, $57 \sim 61$ cores to be specific (Intel. 2013*c*, 2012, 2013*d*). There are, however, four unique features of the coprocessor worth noting. First, the coprocessor has large built-in memory ranging from 6 to 16 GB. Second, each core has a 512-bit wide vector processor (Intel. 2013*e*). The vec-

tor processor adopts the single instruction multiple data (SIMD) execution model. In each clock cycle, it can perform 16 32-bit integer operations, 16 single-precision (32-bit) floating point operations, or 8 double-precision (64-bit) floating point operations. There are three major ways to utilize the vector processors on the Intel Xeon Phi coprocessor. The simplest way is to rely on the automated vectorization performed by Intel's compiler. The code should be compiled with high level of optimizations, typically larger than O2. Due to the conservative nature of the compiler, however, the portion of code that can be vectorized this way is usually very limited, if not nothing at all. The second way is to apply the compiler directives (e.g. `#pragma ivdep`) to a block of code to assist the compiler with vectorization. The compiler directives can be regarded as a promise made by the developers that "the marked code does not have data dependency and the compiler should feel safe and free to vectorize it". The advanced way of using vector processors is to explicitly write the code with vectorization syntax, such as using array notations to manipulate multiple data at the same time (Intel. 2013*b*). Third, 4 "hardware threads" are supported per core on the coprocessor as opposed to 2 hyperthreads per core on the conventional CPU. There are both similarities and differences between the hardware threads and the hyperthreads. On the one hand, in both implementations threads have their private architectural state such as the registers, while sharing the execution resource such as the execution engine and the cache (Intel. 2003, 2013*e*). On the other hand, the hardware thread on a coprocessor is designed for in-order execution that typically requires $2 \sim 4$ threads per core for optimal performance, while the hyperthread is for out-of-order execution, whereby it may be beneficial or detrimental to use more than one threads, usually being case-dependent. Fourth, the coprocessor cores are linked with each other via a ring interconnect (Intel. 2013*e*).

The coprocessor supports a wide variety of programming models, summarized in Table 1.2. Especially, it permits some existing models, such as MPI-OpenMP and MPI-Pthreads, that have already been widely adopted in CPU-based systems. This advantage significantly reduces the effort of code porting.

1.5 Literature Review

Several groups have already applied the GPUs to MC-based photon and electron transport simulations. Badal & Badano (2009, 2011) developed the MC-GPU code for X-ray radiography simulation and radiography dose calculations, and reported a speedup of 110 over the CPU-based MC code, PENELOPE (Baró et al. 1995). Jia et al. (2010, 2011) developed the gDPM code based on the CPU-based DPM that was originally created by Sempau et al. (2000), and observed a speedup of $69.1 \sim 87.2$ over the CPU code for radiotherapy dose calculations. Hissoiny et al. (2011) developed the GPUMCD code for coupled electron-photon transport and reported that for electron transport the speedup factors were 210 and 1200 compared to general-purpose codes DPM and EGSnrc, respectively, while for photon transport the numbers were 20 and 940. From our group, Liu et al. (2012*a*) developed the CPU and GPU-based MC codes for CT organ dose calculation. On a single GPU, the code was found to be 19 times faster than the CPU code and 42 times faster than MCNPX (Pelowitz 2008). The speedup factors were doubled on a dual-GPU system. Jahnke et al. (2012) developed the Geant4-based (Carrier et al. 2004) GMC code and claimed a speedup of 4860 over Geant4 running on one CPU core for Intensity-Modulated Radiation Therapy (IMRT) MC simulations. Chen et al. (2012) developed an MC tool for CT dose calculations using multi-slice CT (MSCT), flat-detector CT (FDCT), and micro-CT scanners, and observed a speedup factor in the range of $40 \sim 50$ using a single GPU compared to a single-core CPU.

Caution should be exercised in understanding and interpreting these impressive speedup factors. Most of the studies only compare the parallel GPU code with the single-threaded CPU code run on a single CPU core. But the multi-core CPU architecture has become the mainstream since 2005 (Moore 2010), and Intel CPUs in particular provide additional parallelism through the hyperthreading technology (HTT) (Intel. 2003). Lee et al. (2010) discovered that for a wide variety of algorithms, by applying multithreading and other appropriate optimization techniques, the performance gap between an Nvidia GTX 280 GPU and an Intel Core i7-960 CPU could be reduced, on the average, to 2.5× only. Thus for a fair comparison of different hardware platforms, it is necessary to parallelize and optimize the code

on each platform to make the most of all the hardware resource available (Liu et al. 2014*a*).

1.6 Objectives

This doctoral research aims to fill the gap created by the rapid development of hardware accelerators in the high performance computing industry and their incompatibility with the existing Monte Carlo particle transport codes. Four primary undertakings as follows are carried out.

1. To develop a new generation of Monte Carlo photon transport code on the heterogeneous computing system. The code shall have three variants designed and fine-tuned for three different parallel platforms on the system: the traditional multi-core central processing units (CPU), the new Nvidia graphics processing units (GPU) and the new Intel Xeon Phi coprocessors. The code shall incorporate a validated CT scanner model and have the capability to handle anthropomorphic phantoms to enable Monte Carlo simulations in CT dose calculations.

2. To verify the developed code against the production code Monte Carlo N-Particle eXtended (MCNPX) in a series of dosimetric benchmark tests. To validate the simulation models against the experimental measurements.

3. To evaluate the computing efficiency, energy efficiency, cost effectiveness, scalability of the developed code, and explore the level of concurrency achievable on the heterogeneous computing system.

4. To apply the developed code to clinical CT dosimetry.

CHAPTER 2
MATERIALS AND METHODS

"C is quirky, flawed, and an enormous success."

—Ritchie, Dennis

This chapter expounds the development process of the photon transport module of our Monte Carlo code named Accelerated Radiation-transport Computations in Heterogeneous EnviRonments (ARCHER) (Xu et al. 2013). This photon transport module is specifically designed for the CT dosimetry application. Another component of ARCHER is the electron transport module for the radiotherapy dosimetry application and is detailed in the publication by Su et al. (2014). There are a total of 9 sections in this chapter. Section 2.1 provides a panoramic view of heterogeneous computing, and pinpoints the specific parallel paradigm we are adopting in ARCHER design. Section 2.2 describes the hardware specifications of our heterogeneous computing system. Section 2.3 explains how the photon transport process is simulated using Monte Carlo methods. Section 2.4 explains how the radiation dose from the CT scan is estimated in ARCHER. Section 2.5 showcases the modelling of CT scanner and patients. Section 2.6 elaborates on the approaches to develop and optimize different variants of ARCHER to specific hardware platforms. Section 2.7 shows the verification and validation tests used to examine the functionality and suitability of ARCHER in CT dose calculations. Section 2.8 describes the method

Portions of this chapter previously appeared as: Liu, T., Ji, W. & Xu, X. G. (2013), Development of GPU-based Monte Carlo code for fast CT imaging dose calculation on CUDA Fermi architecture, *in* 'International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)', Sun Valley, ID, pp. 1199-1210.

Portions of this chapter are to appear in: Liu, T., Du, X., Su, L., Ji, W., Carothers, C. D., Shephard, M. S., Liu, B., Kalra, M., Brown, F. B., Fitzgerald, P. F. and Xu, X. G. (2014), 'ARCHER-CT, an extremely fast Monte Carlo code for patient-specifc ct dose calculations using NVIDA GPU and Intel coprocessor technologies: part I — software development and testing', *Phys. Med. Biol.*. (submitted).

to evaluate the performance of ARCHER, including the computing and energy efficiency. Finally, section 2.9 shows our preliminary effort to apply ARCHER to a clinical CT scan procedure.

2.1 Overview

Developing a code capable of utilizing the emerging hardware accelerators — the GPU and coprocessor — requires the provision of a heterogeneous computing system, defined as the one that uses more than one kind of processors (AMD. 2012). The CPU is still an indispensable component, since the hardware accelerators alone have so far been unable to work independently and need CPU's coordination. Figure 2.1 demonstrates a generic model of such heterogeneous system, which is constructed in a hierarchical pattern with respect to the hardware architectures as well as the programming models. For the hardware architectures, the system contains a number of individual *nodes* connected with one another. Each node has several CPUs and hardware accelerators. Every CPU controls a set of hardware accelerators in that node. For the programming models, on the CPU or coprocessor, there can be one or more processes, each containing a group of threads. On the GPU, one or

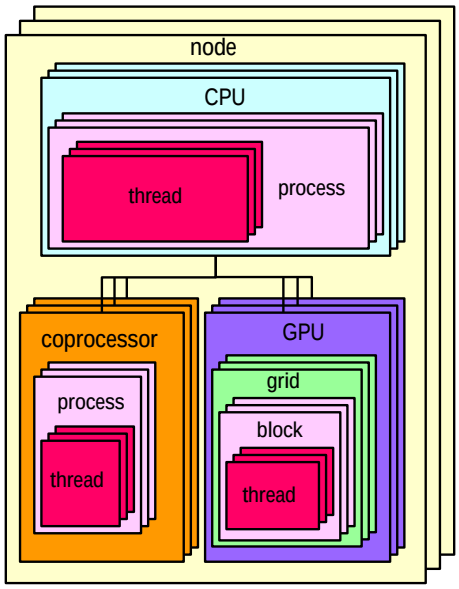


Figure 2.1: The generic model of a heterogeneous computing system. Both the hardware architectures and the programming models are based on the hierarchical design.

more grids can reside, each being made of a number of blocks, which can further be split up into a number of threads.

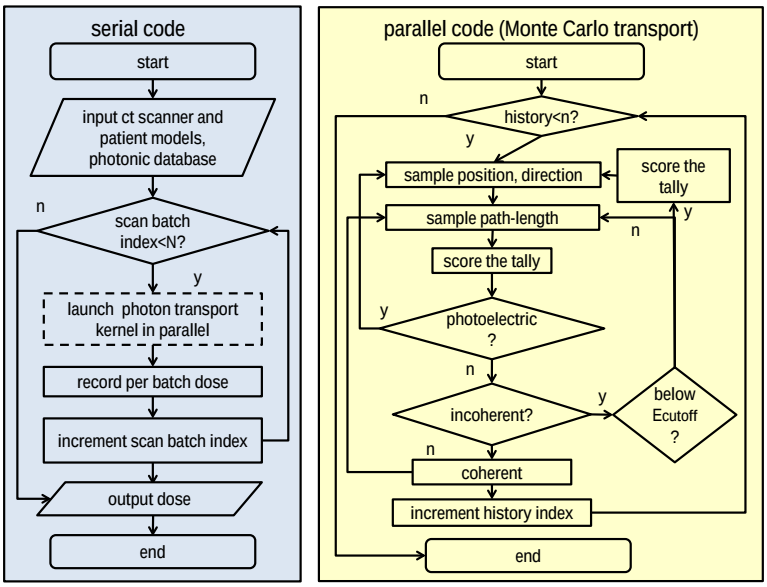
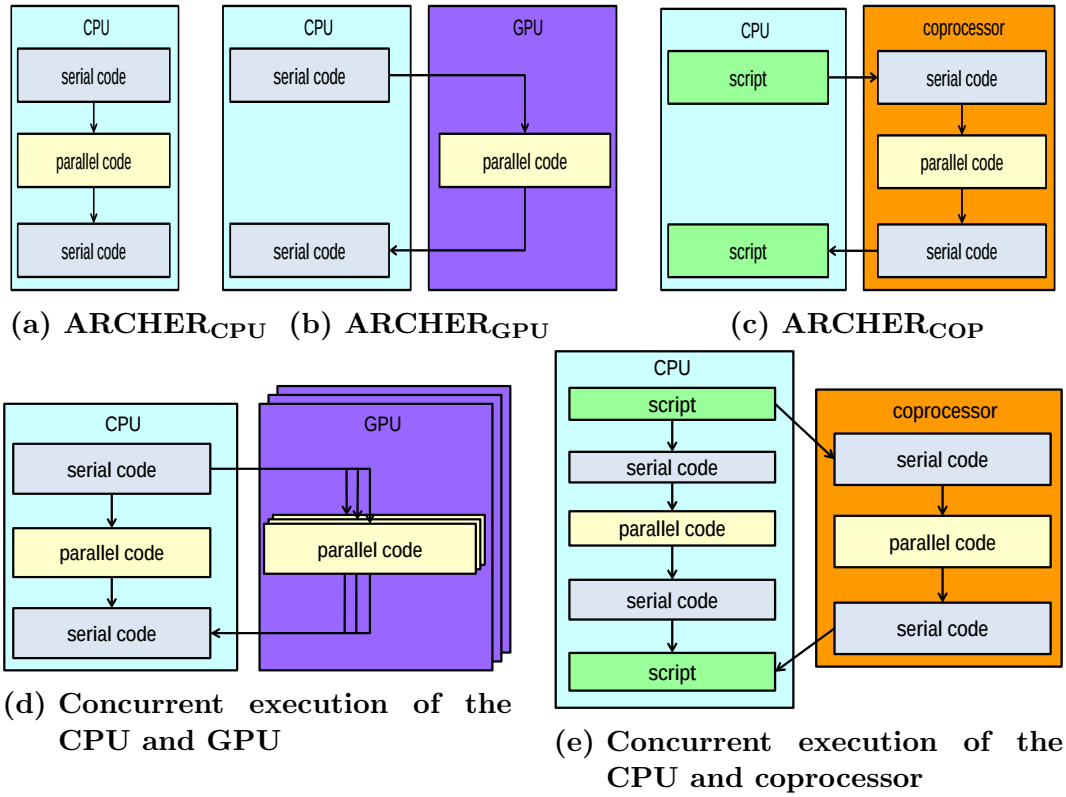


Figure 2.2: General flowchart of ARCHER. The meaning of the symbols are: rounded rectangles — start or end; rectangles with dotted line — indication of the entry point of the parallel code; regular rectangles — generic processing; parallelograms — file I/O; diamonds — decision-making.

This research focuses on implementing a special case of such heterogeneous computing paradigm, whereby a single node is built with a single CPU to control several hardware accelerators, and a new Monte Carlo photon transport code is developed to test the efficacy of each computing unit. The code is named as Accelerated Radiation-transport Computations in Heterogeneous EnviRonments (ARCHER). It is composed of three variants, the CPU code $\text{ARCHER}_{\text{CPU}}$, the GPU code $\text{ARCHER}_{\text{GPU}}$, and the coprocessor code $\text{ARCHER}_{\text{COP}}$. Conceptually, all the codes are composed of two logical parts, shown in figure 2.2, the serial code (in light blue) performing initialization, finalization, file I/O, scheduling, etc, and the parallel code (in light yellow) performing the compute-intensive Monte Carlo simulations.

The parallel part of ARCHER is ultimately what we aim to accelerate and compare across different hardware platforms in terms of the computation performance, energy efficiency, etc. Thus each code variant is named after their specific



(a) **ARCHER$_{\text{CPU}}$** (b) **ARCHER$_{\text{GPU}}$** (c) **ARCHER$_{\text{COP}}$**

(d) **Concurrent execution of the CPU and GPU**

(e) **Concurrent execution of the CPU and coprocessor**

Figure 2.3: Relationship between the serial and parallel parts of ARCHER and the hardware platform. (a) ARCHER$_{\text{CPU}}$ is entirely run on the CPU. (b) ARCHER$_{\text{GPU}}$ has the parallel part run on the GPU and the serial part on the CPU. (c) ARCHER$_{\text{COP}}$ is entirely run on the coprocessor, while the script to manage data transfer between the host and coprocessor executes on the CPU. (d) The CPU and GPU work concurrently. (e) The CPU and coprocessor work concurrently.

hardware platform where the parallel part is physically executed. It is, however, necessary to clarify that for the hardware accelerator variants, a certain part of the code always needs to run on the CPU. The relationship between the code and the hardware platform is more clearly illustrated in figure 2.3. ARCHER$_{\text{CPU}}$ is entirely run on the CPU. ARCHER$_{\text{GPU}}$ has only its parallel part run on the GPU. The entire executable file of ARCHER$_{\text{COP}}$ is run locally on the coprocessor, but the script used to upload the input data and the executable file to the coprocessor as well as download the result back to the host still needs to run on the CPU. Aside from developing each code variant, another important area we have explored is to increase

the system concurrency, i.e. to have multiple computing units running Monte Carlo simulations at the same time to maximize the overall arithmetic throughput. This aspect of work, shown in figure 2.3d and figure 2.3e, is done on multiple scales. In an increasing order, it includes achieving the concurrency of multiple grids on a single GPU, the concurrency of multiple GPUs, and the concurrency of CPU and hardware accelerators.

2.2 Hardware Specifications

The heterogeneous computing system used in this research is built upon a Tyan FT77-B7015 4U Rackmount server (TYAN. 2010). The motherboard has two Land Grid Array (LGA) 1366 CPU sockets and eight second-generation Peripheral Component Interconnect Express slots with 16 lanes (PCIe 2 $\times$16). The computing units used include one Intel Xeon X5650 CPU, six Nvidia Tesla M2090 GPUs, one Nvidia Tesla K20 GPU, one Nvidia Tesla K40 and one Intel Xeon Phi 5110p. The hardware accelerators are mounted to or dismounted from the PCIe slots according to the actual need in different tests. The specifications of the computing units are summarized in table 2.1, where the memory refers to the external main memory for the CPU, and internal onboard memory for the hardware accelerators.

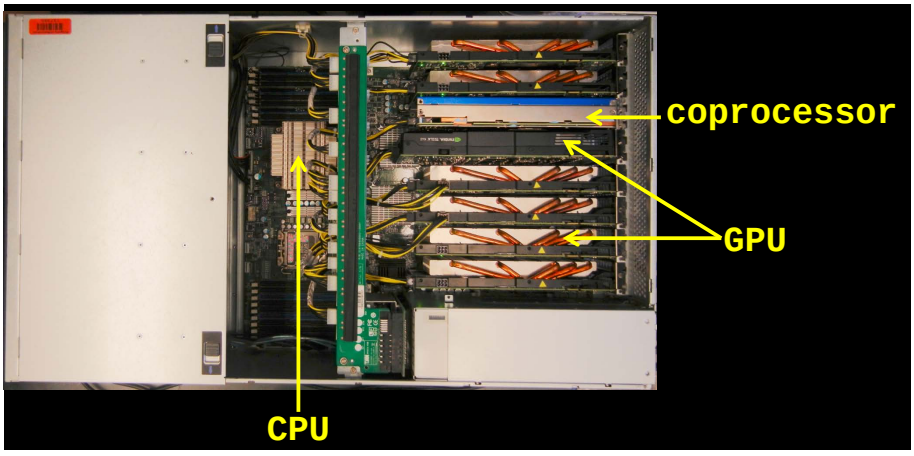


Figure 2.4: A photograph of our heterogeneous computing server.

Table 2.1: Computing units of the heterogeneous computing system.

hardware	model	processor specification	microarchitecture	memory specification
CPU	Intel Xeon X5650	6 cores	Westmere	16 GB
GPU	Nvidia Tesla M2090	16 SMs, 32 SPs per SM	Fermi	6 GB
	Nvidia Tesla K20	13 SMs, 192 SPs per SM	Kepler	5 GB
	Nvidia Tesla K40	15 SMs, 192 SPs per SM	Kepler	12 GB
coprocessor	Intel Xeon Phi 5110p	60 cores	Knights Corner	8 GB

2.3 Monte Carlo Methods

2.3.1 Theory

The Monte Carlo photon transport problem is often centered around quantifying the photo-atomic collision density. It can be mathematically described by the linear time-independent Boltzmann transport equation 2.1 (Brown 2005), where $\mathbf{r}$ is the position vector, $\mathbf{v}$ is the velocity vector, $\Psi(\mathbf{r}, \mathbf{v})$ is the angular collision density, $S(\mathbf{r}', \mathbf{v})$ is the source term, $C(\mathbf{r}', \mathbf{v}' \rightarrow \mathbf{v})$ is the collision kernel that changes the velocity of the particle at a certain position, and $T(\mathbf{r}' \rightarrow \mathbf{r}, \mathbf{v})$ is the transport kernel that changes the position at a certain velocity.

$$\Psi(\mathbf{r}, \mathbf{v}) = \int d\mathbf{r}' \left[\int \Psi(\mathbf{r}', \mathbf{v}') C(\mathbf{r}', \mathbf{v}' \rightarrow \mathbf{v}) d\mathbf{v}' + S(\mathbf{r}', \mathbf{v}) \right] T(\mathbf{r}' \rightarrow \mathbf{r}, \mathbf{v}) \tag{2.1}$$

An alternative form to the Boltzmann transport equation 2.1 with simpler notations and clearer physical meaning is given by equation 2.2 and equation 2.3 (Brown 2005). The position vector $\mathbf{r}$ and velocity vector $\mathbf{v}$ represent the status of the photon, and can therefore be grouped into $p = (\mathbf{r}, \mathbf{v})$; the collision kernel

$C(\mathbf{r}', \mathbf{v}' \rightarrow \mathbf{v})$ and transport kernel $T(\mathbf{r}' \rightarrow \mathbf{r}, \mathbf{v})$ change the status of the photon from p' to p and can therefore be combined into $R(p' \rightarrow p) = C(\mathbf{r}', \mathbf{v}' \rightarrow \mathbf{v})T(\mathbf{r}' \rightarrow \mathbf{r}, \mathbf{v})$. Equation 2.2 indicates that the collision density $\Psi(\mathbf{r}, \mathbf{v})$ can be regarded as a superposition of a series of components having exactly k photo-atomic collisions $(k = 0, 1, 2, ...)$, while equation 2.3 defines each component.

$$\Psi(p) = \sum_{k=0}^{\infty} \Psi_k(p) \tag{2.2}$$

$$\Psi_k(p) = \begin{cases} \int S(\mathbf{r}', \mathbf{v_0}) T(\mathbf{r}' \rightarrow \mathbf{r_0}, \mathbf{v_0}) d\mathbf{r}' & k = 0 \\ \int \Psi_{k-1}(p_{k-1}) R(p_{k-1} \rightarrow p_k) dp_{k-1} & k = 1, 2, ... \end{cases} \tag{2.3}$$

By repeatedly substituting for $\Psi_k(p)$ $(k = 0, 1, 2, ...)$, one finally obtains equation 2.4 (Brown 2005), which pinpoints the fact that the history of a photon consists of a sequence of status transitions, and that the transition is *Markovian*, meaning that status p_k relies only upon p_{k-1} and is irrelevant to prior statuses.

$$\Psi_k(p_k) = \int dp_0 \Psi_0(p_0) R(p_0 \rightarrow p_1) \int dp_1 R(p_1 \rightarrow p_2) ... \int dp_{k-1} R(p_{k-1} \rightarrow p_k) \tag{2.4}$$

This property of photon transport is faithfully simulated by the Monte Carlo methods using random sampling. The process is as follows. At the beginning the initial photon status is $\Psi_0(p_0)$ stochastically determined. Then the next photon status is determined by sampling from the the distribution of the status transition represented by $R(p_0 \rightarrow p_1)$. More concretely, this involves sampling the path-length, i.e. the distance that the photon travels before collision with an atom, from the transport kernel to change the position of the photon, and sampling the interaction type, such as the photoelectric effect, incoherence scattering and coherent scattering for CT X-rays, from the collision kernel to change the energy and flight direction. The random-walk process continues by repeatedly sampling from $R(p_k \rightarrow p_{k+1}), k = 1, 2, ...$ and the photon is free to travel throughout the heterogeneous problem geometry. The history of the photon is terminated once it is absorbed by

the photoelectric effect or escapes from the region of interest.

The physical quantities of interest, such as the flux, current and energy deposition, are tallied and accumulated during the history of a single photon and across many histories. The ensemble estimate obtained yields an expected-value solution of the transport equation (Brown & Martin 1984), demonstrated by equation 2.5, where $x_{j,k}$ is the contribution of each collision to the tally of a single history, x_i is the contribution of each history to the overall tally, and $\bar{x}$ is the expected value.

$$\bar{x} = \frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{\infty} x_{j,k}\right)}{n} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2.5}$$

Due to the statistical nature of the Monte Carlo method, the mean value $\bar{x}$ itself is a random variable and is always accompanied by a statistical error. The mean value along with its error constitutes the Monte Carlo calculation results. The central limit theorem (X-5 Monte Carlo Team 2003*a*) dictates that $\bar{x}$ follows the normal distribution with a variance given by equation 2.6. It is a common practice to report the statistical error in the form of relative standard deviation, defined by equation 2.7, which represents the statistical precision as a fractional result with respect to the estimated mean (X-5 Monte Carlo Team 2003*a*).

$$S_{\bar{x}}^2 = \frac{\sum_{i=1}^{n}(x_i^2)}{n} - \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2 \tag{2.6}$$

$$R = \frac{S_{\bar{x}}}{\bar{x}} \tag{2.7}$$

2.3.2 Radiation Transport Simulation in ARCHER for CT

The Monte Carlo simulation in ARCHER starts at the focal spot of the X-ray tube. The initial position $P = (x, y, z)$ and flight direction $\vec{\Omega} = (u, v, w)$ of the photon are sampled according to the focal spot geometry and the X-ray beam shape, and the initial energy E is sampled from a given kVp-dependent spectrum. The photon is continually tracked throughout the geometries — the analytical model

of the CT bowtie filter, the voxelized patient phantom and the surrounding air — until it is absorbed, leaks the region of interest or falls below the cutoff energy (1 keV). The path-length s is sampled using the Woodcock delta tracking method (Woodcock et al. 1965) to reduce the computation cost. For a photon at a certain energy E, the macroscopic cross-sections of all the cells are calculated and the maximum value $\Sigma_{tt,max}(E)$ is found out. This value is termed *fictitious macroscopic cross-section* in the nuclear engineering. The path-length is then sampled from an exponential distribution using equation 2.8, where ξ is a pseudo-random number uniformly distributed between 0 and 1.

$$s = -\frac{log\xi}{\Sigma_{tt,max}(E)} \tag{2.8}$$

For performance consideration, the fictitious macroscopic cross-section is pre-tabulated instead of being calculated on the fly. Each time the photon is relocated, a "where am I" subroutine is called to find out which geometry the photon is inside of and update the value of the local macroscopic cross-section $\Sigma_{tt}(E)$ accordingly. Then the conditional expression $\xi < \frac{\Sigma_{tt}(E)}{\Sigma_{tt,max}(E)}$ is evaluated. If it evaluates to true, then the photon is considered to have a realistic collision, and the Monte Carlo random walk proceeds. If it is false, then the collision is regarded as "virtual," and the path-length is resampled, and the conditional expression reevaluated, until it becomes true.

Once the realistic collision site is determined, the atom with which the photon collides is randomly sampled. To improve ARCHER's performance, we adopt a semi-deterministic method by regarding the photon interacting with the entire molecule, and avoiding the calculation of atom-specific, energy-dependent microscopic cross-sections. Instead, the macroscopic cross-sections for different types of interactions are pre-tabulated. This photo-molecular interaction model is proven an efficient alternative to the analogue photo-atomic model.

The interaction type is sampled from three possible events dominant in the CT application: photoelectric effects, incoherent scattering (i.e. Compton scattering) and coherent scattering (i.e. Rayleigh scattering) (Attix 2008, pp. 124–160). More concretely, let Σ_{pe}, Σ_{inc}, Σ_{tt} represent the photoelectric effect, incoherent scattering

and total linear attenuation coefficients, respectively. The photoelectric effect will occur if $\xi < \Sigma_{pe}/\Sigma_{tt}$; the incoherent scattering will occur if $\Sigma_{pe}/\Sigma_{tt} \leq \xi < (\Sigma_{pe} + \Sigma_{inc})/\Sigma_{tt}$; and the incoherent scattering will occur otherwise.

In the photoelectric effect, the photon is terminated immediately. In ARCHER, we adopt a special way to simulate the ensuing fluorescence X-ray. If the photon collides with a certain group of organs that contain "fluorescence-prone" atoms, then whether the photon collides with those atoms is randomly sampled according to the cross-section fraction. For atoms with $Z \geq 31$ — which is labelled as the fluorescence-prone atom — the primary and secondary fluorescence are both explicitly simulated (Everett & Cashwell 1973). In our phantom only the iodine ($Z = 53$) located in the thyroid falls into this category, and it releases primary fluorescence with energy as high as ∼30 keV. For $Z \leq 11$, the fluorescence is not simulated because it is below the cutoff energy (1 keV) (Everett & Cashwell 1973). For $12 \leq Z \leq 30$, the primary fluorescence is ignored in the current release of ARCHER.

The scattering interactions are accurately modelled by accounting for the electron's binding effects (Cashwell et al. 1973). Specifically, for incoherent scattering, the angular distribution of the scattered photons is modified by the incoherent form factor $I(Z, \nu)$ to reduce the scattering cross-section in the forward direction. The probability density function (PDF) is expressed by equation 2.9, where μ is the polar angle cosine, Z is the atomic number, E is the energy of the incoming photon, ν is the inverse length ($\nu = \omega E\sqrt{1-\mu}$, ω is a physical constant), $K(E, \mu)$ is the classic Klein-Nishina differential cross-section, and $\sigma_{inc}(Z, E)$ is the incoherent cross-section.

$$f_{inc}(\mu) = I(Z, \nu)K(E, \mu)/\sigma_{inc}(Z, E) \tag{2.9}$$

For coherent scattering, the angular distribution is modified by the coherent form factor $C^2(Z, \nu)$ to reduce the scattering cross-section in the backward direction, whose PDF is expressed by equation 2.10, where $T(\mu)$ is the classic Thomson

differential cross-section and $\sigma_{coh}(Z, E)$ is the incoherent cross-section.

$$f_{coh}(\mu) = C^2(Z, \nu)T(\mu)/\sigma_{coh}(Z, E) \tag{2.10}$$

To improve performance, μ is obtained by sampling from pre-calculated lookup tables based on the two PDFs. The tables are three dimensional matrices $\mu_{i,j,k}$, where i is the material index, j is the energy grid index, and k is the index of the cumulative density function (CDF) grid for f_{inc} or f_{coh}. The method to generate and sample from these matrices is similar to the one devised by Jia et al. (2012).

Secondary electrons from the photo-atomic interactions are not simulated, and electron energy is assumed to be locally deposited. This is a valid assumption because for the CT application the Continuous Slowing Down Approximation (CSDA) range of electrons inside the phantom is generally one order of magnitude smaller than the dimension of a voxel. The capability of electron transport is developed in another module of the ARCHER for radiation therapy (Su et al. 2013).

The photo-atomic data are derived from MCPLIB04 library (X-5 Monte Carlo Team 2003*a*). The pseudo-random numbers in ARCHER are generated using the Xorshift algorithm (Marsaglia 2003) provided by the CURAND library (Nvidia. 2012). Although this generator is not ideal because it does not pass some of the statistical tests (Panneton & L'ecuyer 2005), it is fast (Marsaglia 2003, Nvidia. 2012) and has a good enough quality for Monte Carlo particle transport simulations (Jia et al. 2012, Liu et al. 2014*a*).

2.4 Radiation Dose Calculations

2.4.1 Dose Tallies in ARCHER for CT

The dosimetric quantity of interest in our MC calculation is the absorbed dose D. Under the Transient Charged Particle Equilibrium (TCPE) condition (Attix 2008, pp. 61–80), which is usually satisfied in the human body, it is equal to the collision kerma K_c. ARCHER counts D using MCNP's pre-tabulated heating numbers that represent the average energy deposition per collision (X-5 Monte Carlo Team 2003*a*). The count is carried out prior to the determination of a specific interaction type. The heating numbers include the fluorescence energy for $Z \leq 11$ and exclude

it otherwise. The analytical expression of D is given by equation 2.11, where m is the mass of the tallied organ or tissue, E is the photon energy, t is the time, V is the tallied space, $\vec{\Omega}$ is the solid angle, H is the heating number representing the average energy deposition per collision, $\vec{r}$ is the spatial vector, Σ_t is the total linear attenuation coefficient, also known as the total macroscopic cross-section, and ϕ is the angular flux of the photon.

$$D = \frac{1}{m} \int \mathrm{d}E \int \mathrm{d}t \int \mathrm{d}V \int \mathrm{d}\vec{\Omega}\, H(E)\Sigma_t(\vec{r}, E)\phi(\vec{r}, \vec{\Omega}, E, t) \tag{2.11}$$

In MC method, this quantity is estimated using equation 2.12, where ΔD denotes the dose increment in a single collision event.

$$\Delta D = \frac{H(E)}{m} \tag{2.12}$$

Special treatment was applied to the calculations of dose to the bone surface and red bone marrow (Schlattl et al. 2007). First, since the bone surface is as thin as 10 μm and cannot be directly modelled in voxel phantoms, its dose is approximated by the dose to the spongiosa (Zhang et al. 2009). Second, the dose to the red bone marrow D_{RBM} is derived from the dose response function $R_{RBM}(E)$, an energy-dependent weighting factor, shown in equation 2.13.

$$D_{RBM} = \frac{1}{m} \int \mathrm{d}E \int \mathrm{d}t \int \mathrm{d}V \int \mathrm{d}\vec{\Omega}\, R_{RBM}(E)\phi(\vec{r}, \vec{\Omega}, E, t) \tag{2.13}$$

In MC method, this quantity is estimated using the collision estimator in equation 2.14.

$$\Delta D_{RBM} = \frac{R_{RBM}(E)}{m\Sigma_t(\vec{r}, E)} \tag{2.14}$$

In ARCHER, the above method for dose tallies is implemented in three forms, listed as follows.

- *Organ dose tally* This type of tallies applies to the case where the computational phantoms have well-segmented organs, including all the voxelized phantoms described in section 2.5.2. An organ in such phantoms may contain one

or more cubic voxels that have exactly the same density and elemental composition. The dose and statistical error are calculated for the entire organ.

- *Point dose tally* This type of tallies applies to the case where the phantoms are converted directly from the DICOM image described in section 2.5.3 and do not have well-segmented organs. The dose and statistical error are calculated for a user-specified region — a box that contains one or more cubic voxels that may have different density and elemental composition.

- *Dose distribution tally* This type of tallies applies to both type of phantoms mentioned above. It calculates the dose and statistical error to each individual voxel, and the result is a 3-D matrix having the same dimension with the phantom itself.

In both the organ and point dose tallies, each thread on the CPU, GPU or coprocessor holds one full list of local dose counters. There are two different methods to collect the result from all the threads. One is the *atomic summation*. Once a photon on a thread is terminated, the list of local dose counters are added to a single list of global counters. Here the addition operation is made "atomic" in order to avoid race conditions, where multiple threads compete in performing the read-and-modify operation on the same memory location simultaneously. Atomic summation serializes the participating threads and guarantees all the values carried by them are correctly added together. This method was previously used by Jia et al. (2010), Chen et al. (2012) in the CT dose calculations. The other method is the *parallel summation*. Upon finishing a photon, each thread adds the local dose counters to a thread-private list of global counters. In other words, the number of lists of global counters is not one, but the same with the number of threads. After all the threads complete their computation, an elaborately designed reduction algorithm developed by Nvidia (Harris 2011) is used to efficiently sum up the data in the long list of global counters. The atomic summation has the advantage of smaller of memory usage. None the less, it is offset by the disadvantage that the accuracy of floating point calculation may be negatively affected, which is demonstrated in figure 3.1 of section 3.1.1. Hence in ARCHER, the parallel summation method is always used

for the organ and point dose tallies (Liu et al. 2013, Liu et al. 2014*a*).

For the dose distribution tally, due to the limitation of memory space, it is usually not practical to have each thread hold the local dose counters of a large size. Thus the atomic summation method is used as the last resort. The batch statistics (Romano & Forget 2013) method is adopted in ARCHER to correctly evaluate the statistical error without using local dose counters.

2.4.2 Conversion of Simulated Dose to Absolute Dose

The direct output of ARCHER, similar to MCNPX (X-5 Monte Carlo Team 2003*a*), is normalized to be per starting particle and has an unit of MeV per gram per source particle. In order to have a real dosimetric meaning, the total number of particles N resulting from a certain CT scan needs to be known to convert the result into MeV/g and finally into the conventional form mGy. N can be expressed by equation 2.15, where $I(t)$ is the intensity of tube current, t is the time, T is the total duration the X-ray tube is turned on, λ denotes the number of photons emitted from the X-ray tube per unit of electric charge.

$$N = \lambda \int_0^T I(t)\mathrm{d}t \tag{2.15}$$

Strictly, λ is *only* a function of kVp and tube internal geometry and is independent of other external parameters such as the geometries of collimator, bowtie filter, phantom and the movement of the scanner.

While $I(t)$ and t are both predetermined and can be obtained from the DICOM files, the conversion factor λ needs to be experimentally derived. This work was initially performed by Gu (2010) and later modified by Ding (2012). Their experiments were performed using the standard $CTDI_{100}$ setup, in which an air-equivalent ion chamber was placed at the isocenter without the presence of any phantom, and an axial scan with 100 mAs was performed. The conversion factor is thus obtained by equation 2.16, where D_{abs} is the measured absolute dose and D_{sim} is the simulated dose.

$$\lambda_{100} = \frac{D_{abs}\ [mGy/100mAs]}{D_{sim}\ [MeV/g/\text{source particle}]} \tag{2.16}$$

Compared with λ, the coefficient λ_{100} is more convenient in practice in that it is normalized to a better base — per 100 mAs, and also it inherently accounts for the conversion from MeV/g to mGy.

In general, given a total of P discrete scanner projections, the tube current I_i in mA at the ith projection, the duration Δt in sec of the X-ray tube on when the scanner moves from the ith projection to the $i+1$th, and the MC simulated dose $D_{sim,i}$ at the ith projection normalized by the number of histories N_i emitted from that projection, the absolute dose can be estimated by equation 2.17.

$$D_{abs} = \sum_i D_{abs,i} = \sum_i \left(\lambda_{100} D_{sim,i} \frac{I_i \Delta t}{100} \right) = \frac{\lambda_{100} I_i \Delta t}{100} \sum_i D_{sim,i} \tag{2.17}$$

For tube current modulation, equation 2.17 can be directly applied to calculate the overall absolute dose. Usually the component $D_{abs,i}$ is found useful in studying the angular dependency of the dose. For fixed tube current, the total duration of the X-ray tube on T is often explicitly known, and equation 2.17 reduces to equation 2.18. It should be noted that in this study the CT scanner was modeled in a special way (Gu 2010) in that the sampling of initial source direction (mentioned in 2.5.1) takes the beam collimation into account. This indicates that the collimator is now considered part of the X-ray tube geometry, and thus λ_{100} is also a function of beam collimation beyond kVp and tube internal geometry alone.

$$D_{abs} = \frac{\lambda_{100} I_i T}{100 P} \sum_i D_{sim,i} \tag{2.18}$$

2.5 CT Scanner and Patient Modeling

2.5.1 MDCT Scanner Model

ARCHER incorporates a GE LightSpeed 16 Pro multi-detector CT scanner model. It was reviously developed and validated by Gu (2010) and Ding (2012) using MCNPX code. It is composed of two objects: the source and the bowtie filter. The geometry of the source is illustrated in figure 2.5. An isotropic X-ray point source is placed at the center of a hypothetical sphere S which intersects with a

hypothetical box C called *cookie-cutter*. The name originates from MCNPX (X-5 Monte Carlo Team 2003*b*) where a cookie-cutter cell is used to define geometrically complicated source distribution. The intersection of S and C is a curved surface H which determines the shape of the X-ray beam: if the initial direction vector of the source particle is sampled such that its intersection with S is outside of H, it is rejected and re-sampled; otherwise it is accepted and the MC random walk process continues.

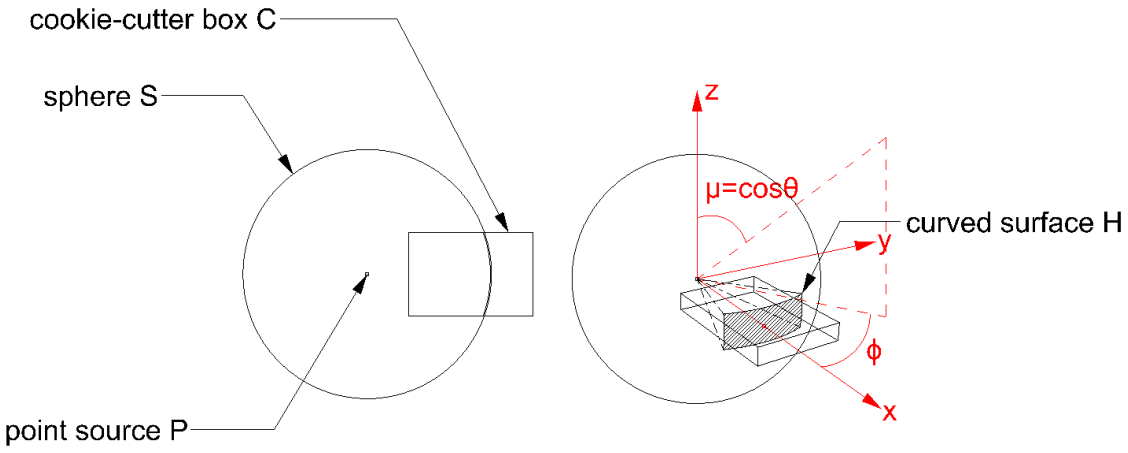


Figure 2.5: Top view (left) and perspective view (right) of the X-ray source model (Gu 2010, Ding 2012).

The area of H is significantly smaller than that of S. Thus, the *isotropic* source model originally defined in MCNPX code results in high rejection rate and unnecessarily long computation time. To alleviate this problem, the source model in ARCHER is modified such that its initial direction is artificially biased toward H. Specifically, in figure 2.5, the vertex of H pointed to by the annotation arrow is the location where the polar angle consine μ (cosine of the angle θ from z axis to the direction vector) and the azimuthal angle ϕ (the angle from x axis to the projection of the direction vector onto x-y plane) achieve their maxima μ_0 and ϕ_0. To effectively reduce the rejection rate, we regulate that $\mu \in [-\mu_0, \mu_0]$ and $\phi \in [-\phi_0, \phi_0]$. It should be noted that for μ and ϕ that are sampled independently following this regulation, the resultant beam area will be slightly larger than H. This is illustrated in figure 2.6. Therefore, to create the intended beam shape, a

rejection sampling process is still needed to filter out the particles whose initial directions point to the green area, but fortunately the rejection rate is considerably lower than the previous isotropic case.

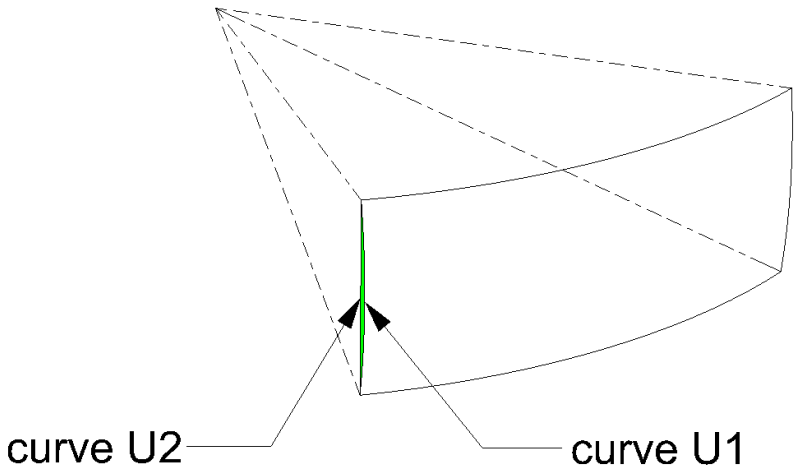


Figure 2.6: Close-up of the X-ray source model. When μ and ϕ are sampled independently, the beam area having U_2 as the boundary is larger than the intersection H of the source sphere and cookie-cutter box having U_1 as the boundary. The difference colored in green is very small, indicating a low rejection rate.

The initial attributes of the particle includes its direction, position and energy. The direction is sampled as stated above. The position is the intersection of H with the direction vector. The energy is sampled from X-ray spectra generated from Xcomp5r, a DOS program developed by Nowotny & Höfer (1985). The anode angle and aluminum flat filter are both taken into account in this utility.

The bowtie filter model is represented by the shaded region in figure 2.7. It is constructed from a box B, two elliptic cylinders E_1 and E_2, and a plane P through boolean operations. It is critical to properly determine the geometric parameters of these objects, as they are directly related to the X-ray beam quality. Because the manufacturer-provided data were unavailable, Gu (2010) adopted an iterative trial-and-error approach to approximate and fine-tune the parameters until the simulation was eventually in agreement with the experiment with 6% tolerance, using the Computed Tomography Dose Index (CTDI) phantom. Gu (2010) modeled two types of bowtie filters for different scan protocols, a head and a body bowtie filter,

which only differ in the geometric parameters.

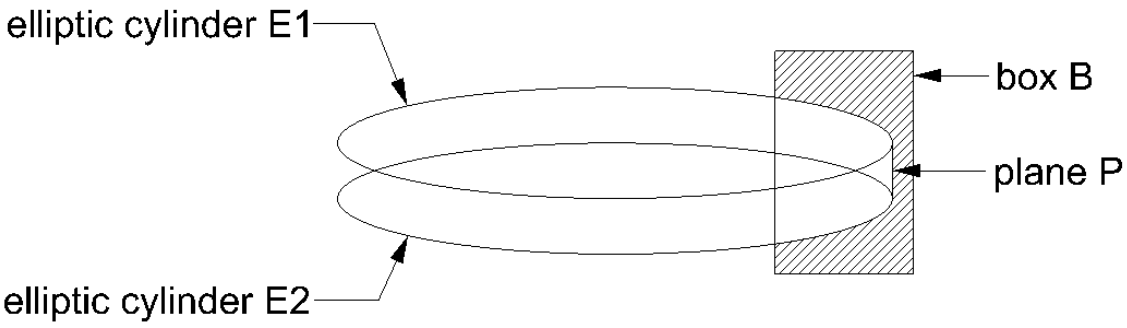


Figure 2.7: Bowtie filer model represented by the shaded region.

The combination of source and bowtile filter models constitutes the MDCT scanner model. In reality, the CT scanner continuously emits X-ray photons when rotating around the patient. In the MC simulation, this process is discretized into a sequence of scanner *projections*: we assume that the scanner moving from position P_i at time t_i to position P_{i+1} at time t_{i+1} delivers the radiation dose that is equivalent to the scanner staying at P_i from t_i to t_{i+1}. The number of projections per rotation is set to 16 as an appropriate approximation suggested by Gu et al. (2009). Two types of scanner movement — axial and helical scans — are modeled. The sign of the incremental angle between adjacent scanner projections is user-specified, and it determines the direction of circular motion (clockwise or counter-clockwise).

With the above scanner model incorporated, ARCHER provides fully customizable scan protocols. Users have the freedom to specify the parameters as follows:

- *Pitch* The pitch is defined as the ratio of the table feed per rotation to the beam collimation.
- *Scan mode* The scan mode encompasses axial or helical scan.
- *Scan region* The region that undergoes CT scan is determined through a pair of parameters: the total number of scanner projections, and the incremental distance along the axis of rotation between adjacent projections. The lat-

ter parameter relates to the pitch as such: pitch = incremental distance × 16/beam collimation

- *kVp* kVp refers to the tube voltage information given in the form of X-ray spectrum. Four pre-tabulated spectra are available: 80, 100, 120 or 140 kVp.
- *Bowtie filter type* Head or body bowtie filters.
- *Beam collimation* The beam collimation is defined as the width of the collimation over the area of active X-ray detection. Four kinds of beam collimation is available to choose from: 1.25, 5, 10, or 20 mm.

Besides, it is noted that the approach by the former researchers in our group to simulate the scanner movement in MCNPX varies from case to case. This complexity has been accounted for in ARCHER by providing three simulation options:

- *Per-projection simulation* The total number of particles is evenly distributed among all the projections, which are treated in sequence: particles emitted from the ith projection are simulated; the radiation doses are calculated and output; then particles from the $i+1$th projection are handled. This option was selected in case of tube current modulation, where the simulated per-projection dose is modified by a projection-specific, angular-dependent coefficient to estimate the realistic dose.
- *Per-rotation simulation* The total number of particles is evenly distributed among all the rotations, which are treated in sequence: particles emitted from the ith rotation are simulated; the radiation doses are calculated and output; then particles from the $i + 1$th rotation are handled. Within each rotation, which projection the particle is emitted from is randomly sampled. This option was selected when a hypothetical whole-body axial scan with pitch of 1 was performed in order to establish a dosimetric database for the VirtualDose software (Ding 2012).
- *Random-selection simulation* The projection from which the particle is emitted is randomly sampled from all probable projections along the scanner trajec-

tory. This option was selected to simulate a clinical helical scan(Ding et al. 2010).

2.5.2 Anthropomorphic Phantoms

ARCHER incorporates a library of voxelized whole-body phantoms with detailed anatomical information. These phantoms were developed in our previous research, including VIP-Man (Xu et al. 2000), RANDO (Wang et al. 2004), RPI-Pregnant women with 3, 6 and 9 months of gestation (Xu et al. 2007), ten extended RPI-Adult females and males representing patients of different Body Mass Indices (BMI), from normal to overweight and to morbidly obese (Zhang et al. 2009, Na et al. 2010, Ding et al. 2012*b*, Liu et al. 2014*a*).

2.5.3 Patient-Specific Phantoms

In a clinical situation, the existing computational phantoms cannot be directly used in MC simulation, due to the fact that their anthropometric and anatomical parameters can significantly differ from those of the real patients. It is necessary to devise a strategy that capitalizes on the clinically obtained information, such as body weight, height or Body Mass Index (BMI), DICOM files, and effectively generates patient-specific phantoms. One proposed approach is to develop in advance a series of voxelized phantoms with incrementally changed parameters and seek the one that best suits the patient. This requires the developed series be sufficiently exhaustive to cover as wide body variations as possible. Examples of such series include RPI deformable phantoms (Na et al. 2010, Ding et al. 2012*b*) and UF adult and pediatric phantoms (Johnson et al. 2009). However, an exact match may rarely happen, whereby the MC simulation using the approximate phantom almost always starts with some existing systematic error. The second possible approach is a variant of the first one. The aforementioned "voxelized" phantoms are in fact derived from their "boundary representation" parent phantoms consisting of a massive amount of control points. The control points provide great flexibility for free geometry alteration not permitted in the voxelized phantoms. Given the patient's anthropometric parameters, a good match can be computationally obtained by interpolating between existing boundary representation phantoms (Liu et al. 2011). However, the

voxelization process that follows, which translates the matching phantom into its voxelized counterpart is very time-consuming — typically 2 days — thwarting the attempt for fast dose calculation.

The third approach adopted in this study has a marked advantage over the former two with respect to the computation speed and matching accuracy. It directly extracts information from the DICOM files, based on which to build up a patient-specific phantom. It is composed of two steps described below.

First, the pixel images and scan protocol information are obtained from the DICOM files. DICOM stands for Digital Imaging and Communications in Medicine. It is a generic, comprehensive standard for storing, handling and transmitting images from a wide variety of medical modalities including CT, computed radiography, magnetic resonance, ultrasound, radio fluoroscopy, etc. A DICOM file is an encoded binary file composed of a sequence of data elements following a common data structure defined by the DICOM specifications (NEMA 1996) and briefly shown in figure 2.8. The first component of the data structure is the "tag" that uniquely identifies the data element. The tag comprises a group tag and an element tag, both being presented as hexadecimal numbers. The second component is the "value representation (VR)" that specifies the data type of the subsequent "value field". For instance, in figure 2.8 "IS" dictates that the data is stored as a string of characters representing an integer in based-10; while "OW" dictates that the data is a string of consecutive 16-bit words. The third component is the "value length" that specifies the size of the subsequent "value field" in byte. The last component "value field" contains the actual data associated with the CT scan. The value field of (0018,1151) indicates that the applied X-Ray tube current is 425 mA, while the value field of (7FE0,0010) contains the entire pixel data of the cross-sectional image. In ARCHER, a DICOM decoding module was developed to carry out this particular step.

Second, the CT numbers from the pixel images are transformed into element weight and mass density data essential for MC simulation (Schneider et al. 2000). Specifically, the CT numbers are first linearly converted to Hounsfield Units (HU) according to the rescale slope and intercept values provided by DICOM. Then the

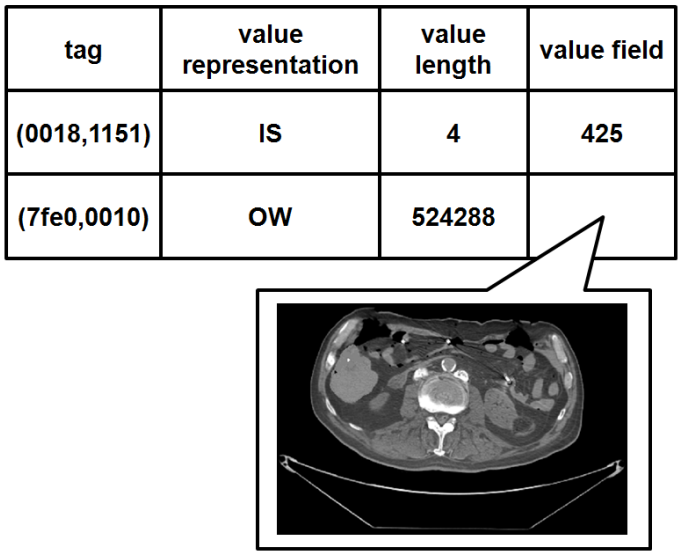


Figure 2.8: DICOM data structure. This example shows two data elements, each consisting of a tag, value representation, value length and value field. The value field of (0018,1151) specifies the intensity of X-Ray tube current, while that of (7FE0,0010) specifies the pixel data array.

material type can be determined from a lookup table (table 2.2) where the scale of HU is subdivided into 24 bins and each corresponds to a unique material with elemental weights already pre-defined.

Table 2.2: Conversion of Hounsfield Unit into material type for Monte Carlo simulation. Each material has a pre-defined list of elemental weights (Schneider et al. 2000).

Hounsfield Units (HU)	material
$HU < -950$	air
$-950 \leq HU < -120$	lung
$-120 \leq HU \leq -83$	soft tissues type 1
$-83 < HU \leq -53$	soft tissues type 2
$-53 < HU \leq -23$	soft tissues type 3
$-23 < HU \leq 7$	soft tissues type 4
$7 < HU \leq 18$	soft tissues type 5

Table 2.2: Continued.

Hounsfield Units (HU)	material
$18 < HU < 80$	mean values of all tissues
$80 \leq HU < 120$	connective tissue
$120 \leq HU < 200$	skeletal tissues type 1
$200 \leq HU < 300$	skeletal tissues type 2
$300 \leq HU < 400$	skeletal tissues type 3
$400 \leq HU < 500$	skeletal tissues type 4
$500 \leq HU < 600$	skeletal tissues type 5
$600 \leq HU < 700$	skeletal tissues type 6
$700 \leq HU < 800$	skeletal tissues type 7
$800 \leq HU < 900$	skeletal tissues type 8
$900 \leq HU < 1000$	skeletal tissues type 9
$1000 \leq HU < 1100$	skeletal tissues type 10
$1100 \leq HU < 1200$	skeletal tissues type 11
$1200 \leq HU < 1300$	skeletal tissues type 12
$1300 \leq HU < 1400$	skeletal tissues type 13
$1400 \leq HU < 1500$	skeletal tissues type 14
$HU \geq 1500$	skeletal tissues type 15

The mass density can be determined likewise (table 2.3) with a different division of HU scale, and within each bin the density is presented as a continuous function.

The above method is efficient in constructing a phantom that matches the patient's geometry within the scan region. However, two factors may affect the accuracy of the dosimetric result. First, the geometry outside of the scan region cannot be derived due to lack of data, and therefore the scattering dose may not be accurately calculated, resulting in a systematic error. Ideally, this can be solved by gluing parts of the existing voxelized phantom to the top and bottom of the new one to generate a whole-body patient-specific phantom. Accurate spatial registration

Table 2.3: Conversion of Hounsfield Unit into mass density for Monte Carlo simulation. The mass density is a piecewise function of HU and it is continuous within each bin (Schneider et al. 2000).

Hounsfield Units (HU)	density
$HU < -1024$	$\rho = 0.001293$
$-1024 \leq HU < -98$	$\rho = 0.001003 \times HU + 1.028$
$-98 \leq HU \leq 14$	$\rho = 0.000893 \times HU + 1.018$
$14 < HU < 23$	$\rho = 1.03$
$23 \leq HU \leq 100$	$\rho = 0.001169 \times HU + 1.003$
$100 < HU < 1525$	$\rho = 0.000592 \times HU + 1.017$
$HU \leq 1525$	$\rho = 1.92$

will emerge as a new challenge. Second, Schneider et al. (2000)'s conversion method itself does not produce perfectly accurate result, as large errors to the carbon and oxygen weights were observed in their study. The work to improve the accuracy based on these two known factors is beyond the scope of this research.

2.6 Software Development

2.6.1 General Flowchart of ARCHER for CT

The general flowchart of ARCHER running on this system is illustrated in figure 2.2. For $\text{ARCHER}_{\text{GPU}}$, the host first imports the patient and CT scanner models as well as the photo-atomic interaction database to the host's main memory, then copies them to the device memory. According to the pre-specified CT scan range, the computation task is divided into a sequence of independent batches. Every batch simulates one scanner rotation where a pre-set number of X-ray photons are tracked in parallel by many threads. Each thread possesses its local counters to register doses from a number of photons assigned to that thread. The per-batch organ doses, represented by $D_k = \sum_i y_{i,k}$ where $y_{i,k}$ is the dose contribution from the ith particle for the kth batch, are then derived from all the per-thread results

by a global reduction operation and copied to the host for temporary storage. The global reduction is realized by a highly optimized algorithm developed by Nvidia (Harris 2011). To calculate the associated statistical uncertainties, another per-batch quantities — the dose squares $T_k = \sum_i (y_{i,k}^2)$ — are counted and summed up in the same manner with D_k. After all the batches are simulated, the total organ doses and the associated statistical uncertainties are calculated on the host. For K batches and n particles per batch, the total relative standard deviation is calculated by equation 2.19.

$$\nu = \sqrt{\frac{\sum_k T_k}{(\sum_k D_k)^2} - \frac{1}{Kn}} \tag{2.19}$$

For $\text{ARCHER}_{\text{CPU}}$ and $\text{ARCHER}_{\text{COP}}$, the entire procedure mentioned above is performed exclusively on the CPU and coprocessor, respectively. The global reduction is performed by the built-in reduction functions provided by the MPI and OpenMP programming models.

2.6.2 Development of $\text{ARCHER}_{\text{CPU}}$ for CT

The parallel CPU code is written in C using the MPI/OpenMP model. The hyperthreading option of the CPU hardware is enabled. This setup ensures that all the hardware resource is fully utilized to achieve the best performance. Compared with the serial CPU code, the parallel one using 6 threads with hyperthreading disabled is found to be 5.98 times faster, while the one using 12 threads with hyperthreading enabled is 8.78 times faster. In the former case the good scalability on the processor level is attributed to multiple threads running simultaneously on different cores, while in the latter case the additional speedup per core arises from the fact that two threads still share the same hardware execution resource but the memory access latency is efficiently hidden through thread switching.

2.6.3 Development of $\text{ARCHER}_{\text{GPU}}$ for CT

The GPU code was written in C using Nvidia's CUDA paradigm. To improve the performance, three issues were carefully considered: the memory usage, execu-

tion configuration and concurrency. The GPU provides several types of memory with different features for data storage. The constant memory is cached, fast but very small. It was therefore used to store a group of physical constants such as the Avogadro constant and elementary electric charge. A fraction of the constant memory was also automatically used by the compiler to store the parameters passed to the GPU kernel. The global memory is slow but large. In addition, it is cached since the advent of Fermi-generation GPU. It was therefore used to hold the large cross-section tables, the phantom and the dose tally data. The texture memory is a special type of global memory. It has a dedicated texture cache and provides hardware filtering that performs linear interpolation in the process of texture fetching (i.e. when the texture memory is read). It was used to store and interpolate the scattering angle tables mentioned in section 2.3. Specifically, the tables were bound to a special form of texture memory called the "layered texture". Each layer contained a 2-D table for a certain material, where each row of that table contained a series of the scatter angle cosine s at a certain energy. To sample μ, first the material index i (the texture layer index) and the energy indices j, $j+1$ (the indices of two adjacent rows of μs) were determined. Then the indices of two adjacent columns of μs k, $k+1$ were sampled to determine which four μs to be used for the bilinear interpolation, which was finally performed by the texture hardware. The shared memory is fast but very small. It was thus only used in the global reduction process to pre-load the dose data from the global memory (Harris 2011). Another type of memory used was the per-SM L1 cache, which was effectively optimized in two ways. First, its size was set through CUDA API to the largest value, 48 KB per SM. Second, its behavior was changed through the compiler option to benefit the global memory access. On the Kepler GPU, it is by default only reserved for the access of local memory, a compiler managed memory to store per-thread data that are too large to fit the registers. The performance of some of the GPUs such as K40 can be improved by forcing it to cache both the local and global memory load — which is in fact the default behavior of the Fermi GPU — using the compiler option "-Xptxas -dlcm=ca" (Nvidia. 2013*d*).

The execution configuration encompasses the specification of proper numbers

of threads per block T and blocks per grid B, as well as the proper setup of the GPU hardware. In $\text{ARCHER}_{\text{GPU}}$, each thread is assigned a maximum of m photons to be simulated one by one in sequence. The lower limit of m is achieved when the number of threads is maximized such that the total size of dose counters are equal to the GPU memory capacity; the upper limit is achieved when the number of threads is minimized such that the GPU achieved occupancy is still sufficiently large, i.e. all the SMs are still busy. For instance, for the simulation of a batch of 10^7 photons and 44 dose tallies on the K40 GPU with 12 GB memory (11.519 GB effective memory for users after ECC is enabled), if each thread only simulates 1 photon, the total memory usage will be:

$$\frac{44 \times 2 \times 4 \times 10^7}{1024 \times 1024 \times 1024} = 3.3\ GB$$

Here "2" refers to the fact that two copies of dose counters are needed: one to count the per-particle dose, the other is to accumulate this per-particle count, and "4" means four bytes due to single-precision floating point. The result is smaller than the amount of effective memory, hence $m_{min} = 1$. Furthermore, the minimum number of threads that narrowly saturates the GPU is:

$$64 \times 32 \times 15 \times 44\% = 13516.8$$

Here "64" is the maximum number of warps per SM, "32" is the number of threads per warp, "15" is the number of SMs per GPU, and "44%" is the GPU occupancy evaluated by the CUDA occupancy spreadsheet (Nvidia. 2013*c*) and confirmed by the profiler (Nvidia. 2013*g*). Therefore $m_{max} = 10^7/13516.8 \approx 739$. We set $m = 100$ in our simulations as an appropriate choice. The total number of threads t was then n/m, where n is the total number of histories per batch. The number of blocks per grid T was derived from the occupancy spreadsheet (Nvidia. 2013*c*). The GPU code was compiled with "-Xptxas -v" to obtain the register usage per thread, with which to query the spreadsheet for a proper value of B that maximized the GPU occupancy (the value was 30% $\sim$ 40%). In our case $T = 64$ or 128 was optimal on both the Fermi and Kepler GPUs. It follows that the number of blocks

per grid B was t/T. The GPU hardware was configured through the management tool "nvidia-smi" (Nvidia. 2013*f*). The "persistent mode" was turned on to greatly reduce the time spent on loading the GPU driver. Moreover, a function called GPU boost (Nvidia. 2013*d*) specific to the K40 GPU was enabled, which increases the clock frequency while keeping the power draw below the upper limit.

The third consideration is the concurrency, referring generally to the ability of a system to perform multiple operations simultaneously (Rennich 2011). Three types of concurrency were investigated: the single GPU stream concurrency, the multiple GPU concurrency and the CPU-GPU concurrency. One a single GPU, because of the random nature of MC simulations, different blocks tend to take different time to complete their jobs. When the simulation is nearing its end, the resident blocks may not suffice to saturate the hardware, leading to a decrease in the GPU occupancy. For a simulation consisting of a sequence of batches, the period of low occupancy can be accumulated to negatively affect the overall GPU performance. This problem can be effectively solved by using the GPU stream. A stream refers to a sequence of commands that execute in order; multiple streams may run concurrently (Nvidia. 2013*b*). We attached different GPU kernels to separate streams, so that when one kernel on a stream was about to finish and did not fully occupy the hardware resource, kernels on other streams could automatically step in and consume the rest of the resource. The second type of concurrency refers to the simultaneous execution of multiple GPUs. Because CT dose calculations are embarrassingly parallel, meaning the threads are executing independently of one another without intensive communications, the multiple-GPU implementation is expected to bring good scalability. We used one CPU thread to control multiple GPUs, and copied the data to and from different GPUs through asynchronous memory operations. A unique seed was assigned to each GPU to generate independent, statistically uncorrelated random number sequences. For a total of K batches, M GPUs and S streams, each GPU was given K/M batches that were organized into $K/(MS)$ iterations, and each iteration involved the concurrent execution of S batches. The third type of concurrency arises from the heterogeneous computation by the CPU and the GPU. These two computing units adopt an asynchronous execution mode in the sense that

once the GPU kernels are launched the control is immediately returned to the CPU, and that the CPU remains idle before hitting an explicitly specified synchronization point. To use the untapped multi-core CPU resource and improve the overall system performance, a CPU MC code written in OpenMP was executed right after the GPU kernel launch. One important issue is to balance the workload between the CPU and GPU such that they finish the computation at approximately the same point. Currently a simplistic method was adopted. For a given pair of CPU and GPU model, a standard MC test — simulation of a whole-body axial scan over the RPI-Adult Male phantom using 9×10^8 photons — was first performed to derive the speedup factor $\overline{\eta_P}$, defined as the ratio of the number of particles simulated by the GPU per second to that by the CPU. Then for K batches, M GPUs and 1 CPU, the simulation was organized into $K/(M\tilde{\overline{\eta_P}}+1)$ iterations, where $\tilde{\overline{\eta_P}}$ denotes the rounding of $\overline{\eta_P}$ to the nearest integer. In each iteration, the CPU simulated one batch, while each GPU simulated $\tilde{\overline{\eta_P}}$ batches using $\tilde{\overline{\eta_P}}$ concurrent streams (Liu et al. 2014*a*).

2.6.4 Development of $\mathrm{ARCHER_{COP}}$ for CT

The coprocessor code was the same with the CPU code, written in C using the MPI/OpenMP model. The three factors described in section 2.6.3 also apply to the coprocessor code. The problem pertaining to memory usage is simpler, mainly because the coprocessor exposes to users a uniform type of memory to store all the input and output data. To reduce the memory allocation cost, the memory page size was explicitly tuned up from 4 KB to 2 MB using a dedicated "huge page" library (Intel. 2013*a*). With regard to the execution configuration, we let the coprocessor work in the native execution mode (Intel. 2013*e*), whereby the executable file, input data and MPI/OpenMP libraries were manually uploaded to the coprocessor, and then the entire code including both the serial and parallel parts was run on it. We issued a total of 60 processes and pinned them to 60 physical cores correspondingly, each having 4 threads bound to the 4 logical cores (i.e. hardware thread). Because the coprocessor did not have the occupancy problem, the task distribution was more straightforward: the photons in a batch were evenly distributed among the all the

60 processes and then among the 4 threads within each process. The concurrency of the CPU and coprocessor was conveniently obtained by using Intel's MPI management tool (Intel. 2014). The CPU-only and coprocessor-only codes were separately compiled, then launched simultaneously by the MPI tool that implicitly took care of the data transfer. The workload was balanced by evenly distributing the photons in a batch among a bunch of processes, and assigning different numbers of processes to the host and device. Specifically, the same standard MC test mentioned above was performed to determine the speedup factor $\overline{\eta_P}$. Since the coprocessor was given 60 processes, the CPU was assigned $60/\overline{\eta_P}$ processes as the appropriate amount of workload.

It should be emphasized that the Intel Xeon Phi coprocessor is best known for its distinctive vector feature, i.e. each core has a vector processing unit (VPU) with 512-bit wide registers for the single instruction multiple data (SIMD) operations (Intel. 2013*e*). However, the conventional history-based MC algorithm adopted in ARCHER has many conditional branches and scattered memory accesses, making it difficult to directly benefit from that feature. There was very limited room for vectorization. One was to rely on the compiler-driven automatic vectorization. The other was the manual vectorization by adding the compiler directives, such as "`#pragma ivdep`" to those loop structures that did not have data dependencies. Both methods only applied to several inner for-loops in the MC transport kernel and did not lead to appreciable performance improvement (Liu et al. 2014*a*).

2.6.5 Development Tools

This section describes the software tools used in the development and test of ARCHER. They fall into four major categories with distinct functions: compiling, scripting, documenting and version controlling.

Firstly, the compiler is in general used to process the source codes in plain text and produce the machine codes in the form of object files. It can also act as a linker by combining all pieces of object files together to generate a working executable file. The set of compilers with which different variants of ARCHER are developed are shown in table 2.4.

Table 2.4: Compilers used to generate different variants of ARCHER.

platform	code	compiler
CPU	serial, openmp	g++
	MPI	mpicxx
GPU	host	g++
	device	nvcc
coprocessor	serial, openmp, MPI	mpiicpc

g++ is an open-source C++ compiler from the GNU Compiler Collection (GCC) family developed in the GNU Project (Stallman 2003). For $\text{ARCHER}_{\text{CPU}}$, it compiles the serial and multithreaded CPU code. For $\text{ARCHER}_{\text{GPU}}$, it compiles the host CPU code and links the object files generated from the host and device. mpicxx is an open-source MPI C++ wrapper coming from the MPI Chameleon 2 (MPICH2) implementation (Gropp 2002). It passes the CPU code of $\text{ARCHER}_{\text{CPU}}$ to the back-end compiler to be processed, in our case, g++, and links the resulting object files to MPI libraries to generate the MPI executable file. nvcc is a proprietary CUDA compiler developed by Nvidia (Nvidia. 2013*e*). It compiles the device codes of $\text{ARCHER}_{\text{GPU}}$ into the object files with GPU-unique format called fat binary, which later are linked by g++ to the host object files. mpiicpc is a proprietary MPI C++ compiler developed by Intel (Intel. 2013*b*). The way mpiicpc works is very similar to mpicxx, in that it serves as a front-end by passing the coprocessor code of $\text{ARCHER}_{\text{COP}}$ to the internal compiling tool, icpc, and performing linkage to enable MPI functions.

Secondly, script interpreter imports and executes user's commands stored in the script as plain text. Table 2.5 listed all such interpreters used in this research for a variety of purposes.

The Make (Stallman et al. 2013) utility serves as a fundamental and flexible platform for source code compilation. On this platform, users explicitly describe the dependency relationships between the source code, intermediate files (such as the

Table 2.5: Script interpreter used for a variety of purposes.

language	purpose
Make	build ARCHER executable from the source code
Bash	organize batch runs in Linux
	organize the profiling of ARCHER
Batch	organize batch runs in Windows
Python	prepare photo-atomic data
	analyze the result of MC simulations and performance profiling
Matlab	prepare phantom data
	visualize dosimetric data

object files) and executable file, from which the executable file is elegantly generated. Besides, on the occasion that changes are applied to some of the source files, Make is able to identify them and determine which intermediate files needs to be updated, instead of unnecessarily recompiling all the source files from the very beginning, thus increasing the compilation efficiency. The Bash (Ramey & Fox 2010) and Batch (Shammas 1993, pp. 1–20) utilities are chiefly used to organize batch runs, which is commonly seen when ARCHER is tested under a combination of different conditions, such as different beam collimations, different kVps, different bowtie filter types, different phantoms, etc. Such test requires a large number of simulation jobs that would collectively take a long time to complete. With Bash and Batch, these jobs can be orderly performed in sequence on one device, or simultaneously on multiple devices to achieve the job-level parallelism. Bash is also extensively used to organize the profiling of the GPU and coprocessor codes for performance analysis. The Python (van Rossum 2014) and Matlab (MathWorks. 1996) utilities significantly facilitates data processing that occurs prior to or subsequent to MC simulation, such as input data preparation and output data visualization.

Thirdly, the documentation generator Doxygen (van Heesch 2008) is used to clearly and conveniently document the source code and create user's and developer's

guide in HyperText Markup Language (HTML) (Berners-Lee & Connolly 1995) format.

Fourthly, the version control system Git (Chacon 2009) is adopted to manage and maintain the source code, which includes reviewing the changes made over time, reverting the code to a previous state, merging the separately developed modules with the previous stable release, etc.

2.6.6 Fair Comparison Considerations

To ensure a fair performance comparison across the three ARCHER variants, the following items have been considered.

- The code is sufficiently parallelized to fully utilize the hardware resource. This is guaranteed by the parallel programming model and the hyperthread CPU function for the CPU code, the stream implementation and carefully determined execution configurations for the GPU code, and the adequate amount of processes and threads for the coprocessor code.

- Error-Correcting Code (ECC) is enabled on the GPU and coprocessor. Although it reduces the memory size and memory bandwidth, the ECC increases the hardware reliability when running a large amount of jobs for a long period of time.

- The same pseudo-random number generators Xorshift are used in all the codes.

- All the codes are highly optimized by applying appropriate compiler options. For example, all are compiled with a high optimization level of -O3. Another example is that to improve the performance, some floating operations are replaced by their faster and less accurate surrogates, realized by the compiler options (Stallman 2003, Nvidia. 2013*e*, Intel. 2013*b*) summarized in table 2.6. The third example is that all have their unique platform-specific compiler options turned on, summarized in table 2.7.

Table 2.6: Compiler options for fast floating point operations.

code	compiler option
$\text{ARCHER}_{\text{CPU}}$	`-ffast-math`
$\text{ARCHER}_{\text{GPU}}$	`-use_fast_math` `-fp-model fast=2`, `-no-prec-div`,
$\text{ARCHER}_{\text{COP}}$	`-no-prec-sqrt`, `-fast-transcendentals`

Table 2.7: Platform-unique compiler options.

code	compiler option
$\text{ARCHER}_{\text{CPU}}$	`-march=native`
$\text{ARCHER}_{\text{GPU}}$	`-gencode=arch=compute_35,"` `code=\"sm_35,compute_35\"`
$\text{ARCHER}_{\text{COP}}$	`-ipo`

2.7 Verification and Validation

2.7.1 Terminology

Although very commonly used, the terminology of verification and validation is not standardized, and needs to be clearly specified (Kleijnen 1995). In this research, we adopt the definition by MCNP6 development team (Pelowitz 2013*b*). According to this definition (Pelowitz 2013*a*), *verification* is a test of the "functionality". It is "generally performed by code developers," and it "involves performing a series of calculations to determine whether a code faithfully solves the equations and physical models it was designed to solve. "Verification may involve comparison to other codes, to analytic benchmarks, or to experiments." In contrast (Pelowitz 2013*a*), *validation* is a test of the "suitability". It is "generally performed by end users," and it "involves a determination of whether the code sufficiently reproduces reality for a particular range of applications of interest.". "Validation may involve assess-

ing the verification problems (to ensure that the end-user application is bounded), comparing calculations to relevant experiments, or performance of scoping studies (to ensure that parameter changes produce expected changes in results)."

In this research, ARCHER as a special-purpose simulation tool has two components. One is the photon transport model developed from scratch according to the theories of photoatomic interactions and Monte Carlo methods. The correctness of this component is what we have aimed to guarantee, and can be tested by comparing ARCHER with a standard code under exactly the same geometry conditions. This test, according to the definition, is a form of verification. The other component of ARCHER is the achievement from previous studies: the built-in CT scanner model (Gu et al. 2009, Gu 2010), and the internal algorithm to generate appropriate computational human phantom from the DICOM images (Schneider et al. 2000). The correctness of this component is examined by comparing ARCHER's simulations with experimental measurements. This test showcases how far the simulation is from the reality, and can be regarded as a form of validation. This classification is appropriate, considering the way MCNP6 differentiates between the shielding validation (simulations versus experiments) and shield verification (the new MCNP6 code versus the previously validated and verified MCNP5 code) (Pelowitz 2013*a*).

2.7.2 Verification of ARCHER for CT with MCNPX

ARCHER is verified against the production code MCNPX version 2.5.0 in four cases of *organ dose* calculations. These cases use the same scan protocols, including whole-body axial scan, 120 kVp, a pitch value of 1:1, but use different computational human phantoms with different anatomies, including the 73 kg RPI adult male phantom, 142 kg RPI adult male phantom, 122 kg RPI adult female phantom and RPI 9-month pregnant female phantom. For each batch (one scanner rotation) 10^7 photons are simulated. The number of batches is $K = h/b$, where h is the height of the phantom and b is the beam collimation. The total number of photons of all batches is sufficiently large to reduce the relative standard deviation to $\sim$0.5% for both ARCHER and MCNP. Information on the phantom geometry is summarized in table 2.8.

Table 2.8: Geometric information of the phantoms used in ARCHER verification. The number of subregions refers to that of the segmented organs/tissues for a given phantom, i.e. the number of "universes" used when the phantom was originally defined in MCNP code.

case	voxel dimension $[mm^3]$	voxel number	number of subregions
73 kg RPI adult male phantom	$0.35 \times 0.35 \times 0.35$	$114 \times 93 \times 509$	128
142 kg RPI adult male phantom	$0.35 \times 0.35 \times 0.35$	$135 \times 132 \times 509$	128
122 kg RPI adult female phantom	$0.35 \times 0.35 \times 0.35$	$136 \times 133 \times 469$	128
RPI 9-month pregnant female	$0.3 \times 0.3 \times 0.3$	$187 \times 142 \times 545$	36

To make the physics models of the two codes consistent, for MCNPX the electron transport is turned off and only the photon transport is simulated (Liu et al. 2014*a*).

2.7.3 Validation of ARCHER for CT with Experiment

ARCHER is validated with the experimental measurements in *point dose* calculations. The validation encompasses two cases, a helical scan over a human cadaver and the ATOM physical phantom respectively, shown in figure 2.9. The scanning was performed by a team of experienced medical doctors, radiologists and physicists at the Massachusetts General Hospital (MGH). In the cadaver study, the human subject was acquired from a non-profit entity Science Care (Zhang et al. 2014). It was an 88 years old male, 183 cm in height and 67.5 kg in weight, died from natural causes (Zhang et al. 2014). Six Thimble chamber dosimeters (Model 10×5-0.6CT and 10×6-0.6CT)) were used to measure the absorbed dose in different deep and superficial organs. They were 21 mm in length and had 0.6 cm^3 active volume (Zhang et al. 2014). The cross-calibration showed that under the

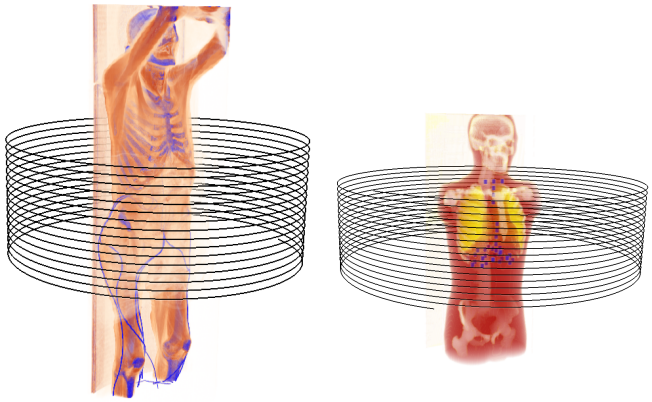


Figure 2.9: CT scan simulations in cadaver (left) and ATOM physical phantom (right) study. The helix around the phantom represents scanner trajectory. This image is generated using the visualization software Paraview (Paraview. 2014).

same kVp, the response from different dosimeters varied within 5%, and that for the same dosimeter, the response under different kVps varied within 0.5% (Zhang et al. 2014). The dosimeters were placed in or next to six organs through surgical procedures (Zhang et al. 2014). These organs included liver, stomach, colon, urinary bladder, left kidney, and paravertebral gutter (Zhang et al. 2014). Several scans were performed over the cadaver using different protocols. Among them, two realistic clinical scan protocols were later simulated. Their similarities included the helical abdomen/pelvis scan, 1 second rotation time, a pitch value of 1.375:1, 16 $\times$ 1.25 mm beam collimation, body-type bowtie filter, a fixed 300 mAs, whereas their difference was the respective kVps — 120 and 100 (Zhang et al. 2014). In addition, a whole-body scan was performed to help construct a complete computational phantom (Zhang et al. 2014). Without it, the scattering dose coming from outside of the scan region in the abdomen/pelvis scan would be neglected, leading to an undesirable dose underestimate.

The direct output data from the experiment are the slice-by-slice reconstructed CT images in the DICOM format mentioned in section 2.5.3. Medical physicists and radiologists at MGH developed semi-automatic algorithms to remove the artifacts

from the images caused by the metal component of the dosimeters. They also computationally determined the exact coordinate of the active volume of the dosimeters from the images.

The DICOM images provided all other scan information that enabled the Monte Carlo simulations. The whole-body CT image data were directly converted into a computational phantom using the procedure mentioned in section 2.5.3. The DICOM tags "trigger on position (0043,1040)" and "duration of X-ray on (0043,104E)" from the abdomen/pelvis scan data were extracted to determine the starting location of the X-ray tube in the x-y plane and the entire scan region along the z direction (including the overscan). The starting location of the X-ray tube in the z direction was determined by image registration for the whole-body and the abdomen/pelvis image data. In the Monte Carlo simulation, the absorbed doses to six $0.49\times0.49\times0.5\ cm^3$ boxes around the dosimeters' active volume were calculated as an estimate of the point dose.

The validation using the ATOM phantom adopted a different approach. The optically stimulated luminescence (OSL) dosimeters were used. The coordinates of the dosimeters were determined from the cross-section photographs of the physical phantom that can be assembled slice by slice. The scan protocols included the helical chest scan, 120 kVp, a pitch value of 0.938:1, 16×1.25 mm beam collimation, and a combination of angular and longitudinal tube current modulation (TCM). An existing computational phantom was directly used. In the Monte Carlo simulation, doses to several $1.4\times1.4\times1.8\ cm^3$ boxes around the dosimeters were calculated. (Gao et al. 2013)

For both validation tests, the Monte Carlo simulations were performed on the Nvidia K40c GPU. The raw results with a unit of MeV/g were converted into mGy/100mAs using the conversion factors obtained in our previous study (Ding 2012) in order to compare with the experimental values (Liu et al. 2014).

2.8 Performance Analysis

2.8.1 Computing Efficiency

2.8.1.1 Performance Comparison of Different Codes

Computing efficiency of different parallel Monte Carlo codes on different hardware platforms is evaluated using a common simulation task — a whole-body axial scan over the 73 kg RPI adult male phantom under 120 kVp and a pitch value of 1:1. A total of 9×10^8 photons are simulated (90 batches, 10^7 photons per batch) to restrict the relative standard deviation to $\sim$0.5% for both ARCHER and MCNP. A scalar quantity — *speedup factor* is defined in equation 2.20 to facilitate the comparison, where $\overline{\eta_P^d}$ is the speedup factor of computing unit d relative to the CPU, N is the floating-point operations count (or equivalently, the number of photons in the simulation) that has been cancelled out, and t_c and t_d are the time of the benchmark test taken by the CPU and the computing unit d, respectively.

$$\overline{\eta_P^d} = \frac{\frac{N}{t_d}}{\frac{N}{t_c}} = \frac{t_c}{t_d} \tag{2.20}$$

2.8.1.2 Performance Comparison with Contemporary Study

The similarity between this research and Chen et al. (2012)'s work in the aspect of GPU-based CT dose distribution calculations allows us to perform a cross-study comparison as an alternative way to evaluate the computing efficiency of $\text{ARCHER}_{\text{GPU}}$. Chen et al. (2012) simulated a CT scan using phantoms with three different spatial resolutions, and listed the corresponding voxel dimensions, number of voxels, number of photons and computation time. According to these data, we modify the dimensions of our abdomen phantom and run the Monte Carlo simulations under similar geometry conditions. Besides, to keep the hardware similar, we choose the Tesla M2090 GPU (ECC on) with a theoretical peak performance of 1331 GFLOPS to compete with the GeForce GTX 285 GPU (without ECC support) with 1063 GFLOPS used by Chen et al. (2012). Considering the fact that the ECC mode negatively affects the GPU performance, it is safe to assume the two GPUs have approximately the same computing power (Liu et al. 2014*a*).

2.8.2 Energy Efficiency

Apart from the computation performance, the energy efficiency is another critical factor, because it is directly associated with the total operating cost and environmental impact of a computing platform. In this study it is evaluated by running a standardized Monte Carlo test on different platforms. The test simulates a single-rotation axial scan over the abdomen region of the 73 kg adult male phantom. It should be pointed out that here we specifically focus on calculating the energy efficiency of each computing hardware for the parallel Monte Carlo transport subroutine alone, and neglect the energy consumption by the serial portion of ARCHER code such as the File I/O and data pre-processing, and by other hardware components such as the system memory and cooling device. Besides, the energy consumption by the idle CPU when the GPU or coprocessor is being tested is ignored. Accurate measurement of the overall system power draw for the whole run is of equal importance and can be done by directly connecting the server to the external power meters, but this is beyond the scope of discussion here.

Currently there has not been a commonly accepted and adopted metric for energy efficiency quantifications. Thus several quantities as follows are investigated together (Liu et al. 2014*a*).

- *Power* The power is defined as the energy per unit of time, a measure of instantaneous energy consumption. We used the platform-specific, software-based monitoring tools to poll the hardware and sample the power at regular intervals while ARCHER is running. The GPU platform provides two command-line utilities to do that. One is nvidia-smi (Nvidia. 2013*f*) for GPU hardware administration, and the other is nvprof (Nvidia. 2013*g*) for GPU code profiling. We chose nvidia-smi instead of nvprof, because the latter appears to be more intrusive to the GPU and can cause a slight power overestimate. The coprocessor platform also provides two utilities. One is micsmc (Intel. 2013*g*) to log the performance, temperatures, and core usage of the coprocessor. The other is the Performance Application Programming Interface (PAPI) (Dongarra 2013). We adopted micsmc rather than PAPI, since the latter has a low time resolution on the order of second only. Currently our CPU platform

based on the Westmere microarchitecture lacks an effective software solution for power monitoring. Therefore we use the benchmark result provided by SPEC. (2014) as a legitimate power estimate — 129 Watts for 100% CPU utilization and 40 Watts for the idle CPU.

On the other hand, according to Hennessy & Patterson (2012), the peak power is often 1.5 times higher than the Thermal Design Power (TDP). It is thus justifiable to use TDP directly as the CPU power estimate, which will lead to a conservative comparison of the energy efficiencies of the hardware accelerators and the CPU.

- *Energy* This metric reflects the total energy demand of the computing unit for a given task, regardless of its performance. It was calculated by integrating the sampled power data over time.

- *FLOPS per Watt* The FLoating-point Operations Per Second (FLOPS) is a measure of the performance of a given computing unit, widely adopted by the high performance computing industry, including Top500. (2013). The FLOPS per Watt metric takes the power issue into account, and is adopted by Green500. (2013) as an essential index of energy efficiency. In this study, the FLOPS of 32-bit single precision was measured using the GPU profiling tool nvprof (Nvidia. 2013*g*), and the average power is calculated by dividing the total energy consumption by the computing time. Due to lack of software tools, the GPU's FLOPS was used as a rough approximation to that of the CPU and coprocessor. Note that despite the use of FLOPS as a factor, this metric in fact does not reflect the platform performance, as is explained in equation 2.21, where η_E is FLOPS per Watt, R is the FLOPS, P is the power, N is the floating-point operations count, E is the energy consumption, and t is the time of the benchmark test that has been cancelled out.

$$\eta_E = \frac{R}{P} = \frac{\frac{N}{t}}{\frac{E}{t}} = \frac{N}{E} \tag{2.21}$$

A computing unit that takes a long time to finish a given task but consumes

a little energy may still have high FLOPS per Watt, although such hardware device is hardly of practical value. It follows that the computing performance and FLOPS per Watt are two separate factors, both of which should be considered when evaluating the efficacy of a computing platform.

2.8.3 Cost Effectiveness

While the computing and energy efficiency of a computing platform are two orthogonal, important considerations, they both boil down to economy: high computing-efficiency amounts to high productivity and profit, while high energy-efficiency to low electricity expense. Ultimately, an economy-related, unified factor is almost always desired to assist with the *buying decision process*. Here we establish a simplistic cost model for our heterogeneous computing system. For *a single 4U rackmount server*, we define the cumulative cost $C(t)$ in unit of US dollar ($) in equation 2.22, where $C_u(t)$ is the cost unique to the given computing unit (the CPU, GPU, coprocessor, etc), and $C_s(t)$ is all the cost other than the computing unit.

$$\begin{cases} C(t) & = C_s(t) + C_u(t) \\ C_i(t) & = C_i^{CA}(0) + C_i^{OP}(t), i = s, u \end{cases} \tag{2.22}$$

Both $C_s(t)$ and $C_u(t)$ contain two parts, where $C_i^{CA}(0)$ refers to the capital cost, i.e. the one-time expenses on the equipment purchase, and $C_i^{OP}(t)$ refers to the operating cost, i.e. the continuous expenses on the server operations. The breakdown of each component is enumerated as follows.

- $C_s^{CA}(0)$ Chassis, motherboard, power supply, cooling device (fan), chipset, memory, hard drive, software (operating system, development kit), site infrastructure, others.
- $C_s^{OP}(t)$ Electricity (consumed by all the devices other than the computing unit), network, labor (unrelated to the computing unit), others.
- $C_u^{CA}(0)$ Computing unit.

- $C_u^{OP}(t)$ Electricity (consumed by the computing unit), labor (related to the computing unit, such as code porting, staff training).

It should be noted that $C_s(t)$ may be similar to or well larger than $C_u(t)$. Consequently, it may undesirably obfuscate our comparison between various parallel computing platforms, where the difference lies only in the computing unit alone. Thus we argue that for the simplicity and clarity of cost analysis, only $C_u(t)$ needs to be taken into account. We herein define a scalar quantity called *normalized cost effectiveness factor* in equation 2.23 to carry out this analysis, where $\overline{\eta_C^d(t)}$ refers to the normalized cost effectiveness factor for the computing unit d, $\overline{\eta_P}$ denotes the speedup factor in section 2.8.1.1, and $C_u^c(t)$ and $C_u^d(\frac{t}{\eta_P})$ are the cost unique to the CPU and the computing unit d, respectively.

$$\begin{aligned}\overline{\eta_C^d(t)} &= \overline{\eta_P} \cdot \frac{C_u^c(t)}{C_u^d(\frac{t}{\eta_P})} \\ &= \overline{\eta_P} \cdot \frac{C_u^{CA,c}(0) + C_u^{OP,c}(t)}{C_u^{CA,d}(0) + C_u^{OP,d}(\frac{t}{\eta_P})}\end{aligned} \tag{2.23}$$

Note that the computing unit d with a speedup factor of η_P can effectively reduce the computation time from t (referring specifically to the CPU wall time) to $\frac{t}{\eta_P}$. Hence for a fixed amount of computation task, the operating cost can also be reduced accordingly. Further simplifications to equation 2.23 are necessary, due to the unavailability of the labor cost data for this study. It is thus assumed that the operating cost only comprises the electricity component, and the following approximation is obtained in equation 2.24, where κ denotes the electricity rate in unit of dollars per Joule, $\overline{P_c}$ and $\overline{P_d}$ refer to the average power draw derived by the method in section 2.8.2.

$$\overline{\eta_C^d(t)} = \overline{\eta_P} \cdot \frac{C_u^{CA,c}(0) + \kappa\overline{P_c}t}{C_u^{CA,d}(0) + \kappa\overline{P_d}\frac{t}{\eta_P}} \tag{2.24}$$

2.8.4 Profiling

To investigate and understand whether $\text{ARCHER}_{\text{GPU}}$ has exploited the full potential of the GPU hardware, additional performance analysis is conducted. The instruction and the memory statistics are gathered by the GPU profiler "nvprof"

(Nvidia. 2013*g*) for the K40 GPU without using the boost function. The computing task is the simulation of a single axial scan over the RPI-Adult Male 73kg phantom in the abdomen region — 1 single batch with 1×10^7 photons. This amount of computing task is appropriate such that it fully saturates the GPU resource while it does not overflow the hardware counters.

2.9 Clinical Applications

The performance of $\text{ARCHER}_{\text{GPU}}$ is assessed in a clinical case at Massachusetts General Hospital (MGH). $\text{ARCHER}_{\text{GPU}}$ was used to simulate an abdominal scan over a patient stricken with prostate cancer and calculate the 3-D absorbed dose distribution. i.e. the absorbed dose to each voxel is separately quantified. The scan protocol included a helical scan mode, 120 kVp tube voltage, 250 mAs per rotation, 16×1.25 mm beam collimation and a pitch value of 1.375:1. A radiation oncologist outlined the tumor target (prostate) and the structures in its vicinity (rectum, bladder and femoral head) on the CT images. The stack of images were then converted into voxelized abdomen phantom for Monte Carlo dose calculations (Ding et al. 2010). The phantom has 218×126×60 voxels with voxel dimensions of 0.1954×0.1954×0.5 cm^3. A total of 10^8 photons were simulated in the dose distribution calculation, which is sufficient to make the statistical uncertainty below 1% in the case of organ dose calculation (Liu et al. 2014*a*).

CHAPTER 3
RESULTS AND DISCUSSION

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

—Turing, Alan

This chapter presents the results of ARCHER development. Section 3.1 demonstrates how much ARCHER is consistent with the production code MCNPX in the verification tests, discusses how different summation strategies impact the accuracy of the GPU code, and illustrates the degree of agreement between the simulated and the realistic geometry models of ARHCER in the validation tests. In section 3.2, the computing and energy efficiency of different parallel Monte Carlo codes on different hardware platforms is compared, accentuating the performance advantage of the hardware accelerators. Finally, section 3.3 shows the preliminary performance result of applying ARCHER to a clinical CT dosimetry case.

3.1 Verification and Validation

3.1.1 Verification of ARCHER for CT with MCNPX

ARCHER is verified against MCNPX version 2.5.0 in organ dose calculations using four different heterogeneous phantoms. Sufficiently many particles are simulated to ensure that the relative standard deviation is restricted to 0.5% for both

Portions of this chapter previously appeared as: Liu, T., Ji, W. & Xu, X. G. (2013), Development of GPU-based Monte Carlo code for fast CT imaging dose calculation on CUDA Fermi architecture, *in* 'International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)', Sun Valley, ID, pp. 1199-1210.

Portions of this chapter are to appear in: Liu, T., Du, X., Su, L., Ji, W., Carothers, C. D., Shephard, M. S., Liu, B., Kalra, M., Brown, F. B., Fitzgerald, P. F. and Xu, X. G. (2014), 'ARCHER-CT, an extremely fast Monte Carlo code for patient-specifc ct dose calculations using NVIDA GPU and Intel coprocessor technologies: part I — software development and testing', *Phys. Med. Biol.*. (submitted).

ARCHER and MCNPX. The results of three ARCHER variants only have negligible differences primarily caused by different number of threads used at runtime. Thus for brevity here we only demonstrate the results from $\text{ARCHER}_{\text{GPU}}$. The data are listed in appendix A. The statistical information is summarized in table 3.1. This level of agreement between ARCHER and MCNPX is considered excellent in the CT dose calculations.

The slight difference between the results by ARCHER and by MCNPX is attributed to four factors. First, the pseudo-random number generators are different. MCNPX adopts Lehmer 48-bit linear congruential generator(LCG) (X-5 Monte Carlo Team 2003*a*,*b*), while ARCHER uses the 32-bit Xorshift generator (Nvidia. 2012). Second, MCNPX models the Doppler broadening effect and the X-ray fluorescence for $12 \leq Z \leq 30$, while ARCHER does not. This is the only difference between the interaction models of the two codes. Third, for efficiency consideration, ARCHER replaces on-the-fly calculations with linear interpolation from lookup tables in several subroutines, such as the calculation of fictitious macroscopic cross-section and the incoherent/coherent scattering polar angle. The interpolation errors may affect the results. Fourth, ARCHER uses the single-precision floating point while MCNPX uses the double-precision (Liu et al. 2014*a*).

One important factor associated specifically with the accuracy of $\text{ARCHER}_{\text{GPU}}$ is the way to collect the dose results from each individual thread and sum them up to get a final answer. As is mentioned in 2.4, both atomic and parallel summation methods have their particular advantages over each other. For the former method, however, as the problem size scales up, the numerical error can become increasingly noticeable, as illustrated in figure 3.1. In this test case, the 73 kg RPI adult male phantom is used, and dose to the red bone marrow from a single axial scan around the abdominal region is calculated. Using the single-precision floating point format, the atomic summation gradually 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 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 less than 0.4% different from MCNPX. In this

Table 3.1: Comparison of the dosimetric results calculated by ARCHER and MCNP in terms of the percentage difference, defined as |ARCHER − MCNP|/MCNP. The absolute value is used to avoid an overestimate of the accuracy where the positive and negative differences accidentally cancel out.

case	total number of photons simulated	maximum difference [%]	minimum difference [%]	average difference [%]
73 kg RPI adult male phantom	9×10^8	0.87	0.03	0.29
142 kg RPI adult male phantom	9×10^8	1.67	0.01	0.31
122 kg RPI adult female phantom	8.3×10^8	1.03	0.00	0.42
RPI 9-month pregnant female	8.2×10^8	1.92	0.04	0.37

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 (Liu et al. 2013).

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 (Nvidia. 2013*b*). This emulation method, however, considerably reduces the overall computational performance and is not feasible in fast dose calculations (Liu et al. 2013).

3.1.2 Validation of ARCHER for CT with Experiment

The results of the dosimetric comparison are shown in figure 3.2, table 3.2 and table 3.3. The relative standard deviation (RSD) in the experiment is 5% according to the manufacture-provided data (Radcal. 2014) for the cadaver case, and is calculated from multiple OSL dosimeter readings for the ATOM phantom case, while that in the Monte Carlo simulation is the statistical uncertainty of the

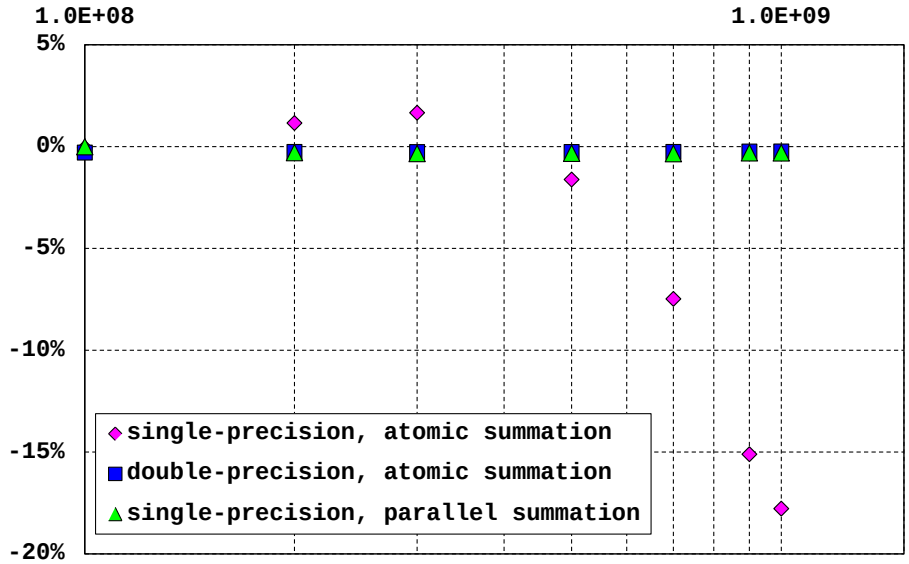
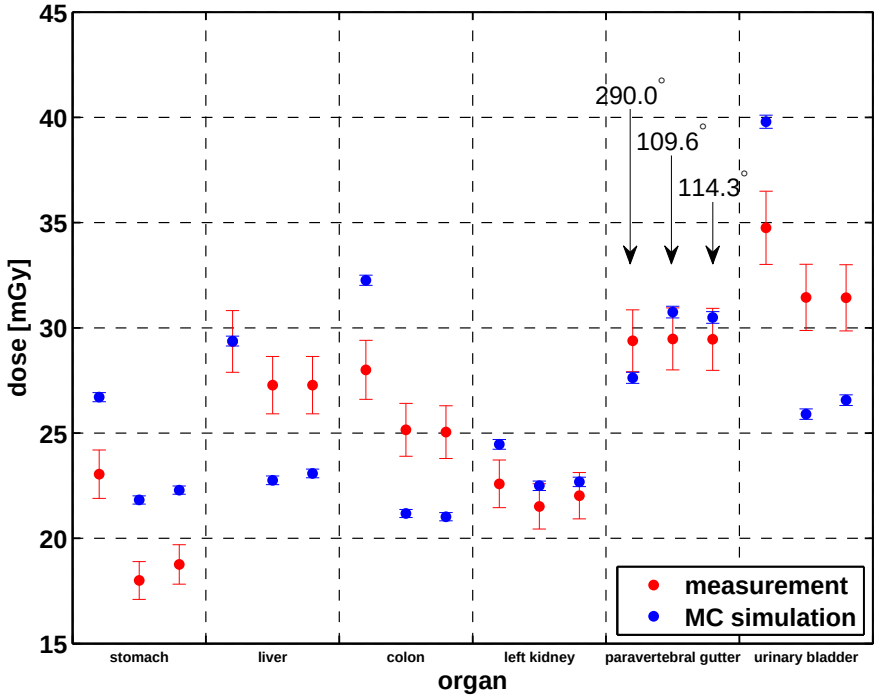
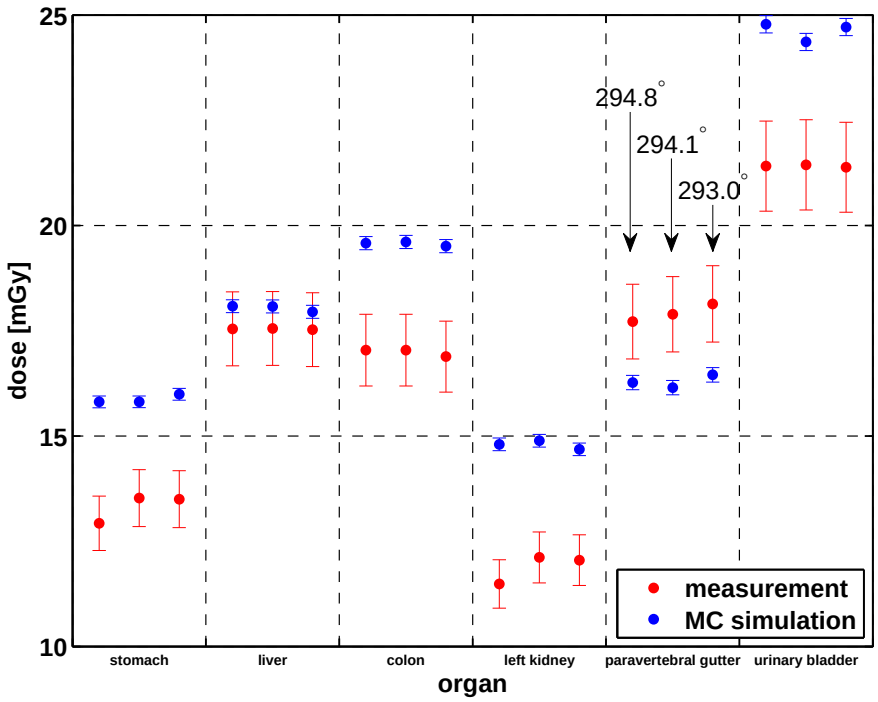


Figure 3.1: The influence of parallel and atomic summation methods over the accuracy of ARCHER$_{GPU}$ in organ dose calculations. This test simulates a single axial scan over the abdominal region using the 73 kg RPI adult male phantom.

mean values. It is found from the output DICOM files that even if scan protocols are kept exactly the same, the X-ray tube starting position ("trigger on position (0043,1040)") in the three measurements is always a random value, annotated in the figure 3.2. The dose difference in the cadaver and ATOM case is within 29% and 40%, respectively. It should be pointed out that the experimental measurement of the dose to the esophagus in the ATOM phantom case is very unreliable, because firstly, only 1 OSL dosimeter was planted in the very beginning slice of ATOM phantom in the experiment, secondly that slice was not completely covered by one scanner rotation, and thirdly the modulated tube current applied was low (120 mA on the average). If this data is excluded, then the dose difference would become within 16%.



(a) 120kVp.



(b) 100kVp.

Figure 3.2: Validation of ARCHER with the experimental measurement using the human cadaver, (a) 120kVp and (b) 100kVp, fixed 300mA tube current.

Table 3.2: Validation of ARCHER with the experiment using the human cadaver, 120kVp, fixed 300mA tube current. The X-ray tube starting position ("trigger on position (0043,1040)") is a random value even when the scan protocols are kept exactly the same.

kVp	120			100		
organ	difference [%]					
starting location [°]	290	109.6	114.3	294.8	294.1	293
stomach	15.88	21.26	18.83	22.32	16.9	18.47
liver	0.06	-16.56	-15.38	3.06	2.98	2.41
colon	15.19	-15.8	-16.05	14.91	15.08	15.54
left kidney	8.31	4.57	2.99	28.88	22.88	21.84
paravertebral gutter	-5.98	4.33	3.52	-8.18	-9.73	-9.29
urinary bladder	14.51	-17.63	-15.47	15.74	13.63	15.58

Table 3.3: Validation of ARCHER with the experiment using the ATOM physical phantom, 120kVp, tube current modulation.

organ	experiment		ARCHER		difference [%]
	dose [mGy]	RSD	dose [mGy]	RSD	
lung	13.8	15.7	11.90	0.01	-13.7
thyroid	14.2	37.1	16.25	0.11	14.4
esophagus	10.1	NA	14.06	0.21	39.2
heart	12.8	7.4	12.10	0.10	-5.5
stomach	11.6	10.2	9.74	0.12	-16.0
liver	13.0	10.3	12.37	0.05	-4.9
spleen	12.2	10.3	10.54	0.17	-13.6
kidneys	11.9	4.7	11.57	0.16	-2.8
thymus	18.8	NA	15.80	0.21	-15.9

The relative standard deviations of both experiments and simulations are a measure of the precision, not accuracy of the dose presented. The observed difference is attributed to the inherent systematic errors that affect the accuracy. ARCHER contains more sources of systematic errors, not in the photon transport models that have been verified with MCNPX, but in the geometry modelling. The most significant error may come from the CT scanner model. Using the $CTDI_{100}$ phantom, 20 mm beam collimation and 100 and 120kVp, a difference of 0.27% $\sim$ 5.22% between the experiments and simulations was reported in the previous study (Ding 2012). Recently, we have simulated the half value layers and compared against the experiments in the isocenter. The differences are shown in table 3.4. This indicates that there is still room for improving our CT scanner model. A second contributory factor lies in the algorithm that converts the DICOM images to a patient-specific phantom. The analytically derived densities and elemental weights may occasionally have large deviation from the true values, specifically for the weight fraction of carbon and oxygen atoms (Schneider et al. 2000). The systematic errors in the experiment is believed to be relatively small and can come from the dosimeter system (Liu et al. 2014, Zhang et al. 2014). Eventually, it is worth reiterating that the observed discrepancy between ARCHER and experiment does not indicate a flawed physics model developed in *this* study — which contrarily has been proven very accurate, but simply suggests that the approach established by *previous* study to simulate the experiment is not ideal and can be improved.

Table 3.4: Comparison of half value layers (HVL) in the isocenter by experimental measurements and Monte Carlo simulations using GE LightSpeed 16 Pro CT scanner.

kVp	experiment [mm]	ARCHER [mm]	difference
80	5.30	4.95	-6.58%
100	6.48	6.13	-5.39%
120	7.52	7.14	-5.07%
140	8.43	8.03	-4.76%

3.2 Performance Analysis

The performance of the developed code is evaluated from two perspectives. One is the conventional factor — computing efficiency, i.e. how much time the code needs to take to complete a given simulation task. The other is the new factor introduced by the modern parallel computing industry — energy efficiency, i.e. how much energy it needs for a given task. These two factors are orthogonal, in that a code that appears very fast on a many-core parallel processor may undesirably demand a large amount of energy and therefore a high electricity cost, whereas a code that appears very energy-saving on a certain platform may take unsought, extremely long time.

3.2.1 Computing Efficiency

3.2.1.1 Performance Comparison of Different Codes

The execution time of different parallel Monte Carlo codes is listed in table 3.5. All the ARCHER variants are found to be computationally efficient and are substantially faster than the parallel MCNPX running with 12 MPI processes. There are three major reasons: first, MCNPX used in this research is a pre-compiled executable with O1 optimization level and double-precision floating point, while ARCHER uses more aggressive optimizations and single-precision format. Second, there are three major differences in the algorithm. (1) ARCHER adopts an improved method for biased source sampling, in which the initial photon position is bounded by the slot created by the "cookie cutter" object mentioned in section 2.5.1, leading to a higher acceptance rate in the rejection sampling process. (2) The Woodcock delta tracking method used in ARCHER for path-length sampling is an efficient alternative to the conventional surface-to-surface ray-tracing adopted by MCNPX. (3) To increase the speed, ARCHER uses the lookup table interpolation instead of the on-the-fly rejection sampling used in MCNPX. Third, MCNPX is a general-purpose production Monte Carlo code that supports many applications outside of medical physics. In contrast, ARCHER is developed specifically for CT simulations, and has been optimized for that specific class of computational models.

Table 3.5: Computation time of different Monte Carlo codes running on different hardware architectures for a whole-body CT scan simulation. A total of 9×10^8 photons are simulated (90 batches, 1×10^7 photons per batch). The numbers in brackets are the speedup factors ($\overline{\eta_P}$) compared to $\mathbf{ARCHER_{CPU}}$. P=MPI processes, T=threads, S=GPU streams.

code	hardware	condition	execution time [min] ($\overline{\eta_P}$)	FOM improvement
parallel MCNPX	X5650 CPU	12 P	476.35	baseline
$\text{ARCHER}_{\text{CPU}}$	X5650 CPU	1 P, 12 T/P	11.22 (baseline)	21.03×
$\text{ARCHER}_{\text{GPU}}$	M2090 GPU	15 S	2.08 (5.40 ×)	113.46×
$\text{ARCHER}_{\text{GPU}}$	M2090 GPU×6	15 S	0.37 (30.23 ×)	635.65×
$\text{ARCHER}_{\text{GPU}}$	K20 GPU	15 S	1.75 (6.40 ×)	134.65×
$\text{ARCHER}_{\text{GPU}}$	K40 GPU with boost	15 S	1.03 (10.89 ×)	228.90×
$\text{ARCHER}_{\text{CPU+GPU}}$	X5650 CPU and K40 GPU with boost	CPU: 1 P, 12 T/P, GPU: 15 S	0.95 (11.85 ×)	249.11×
$\text{ARCHER}_{\text{CPU+GPU}}$	X5650 CPU and M2090 GPU ×6	CPU: 1 P, 12 T/P, first 5 GPUs 5 S, last GPU 4 S	0.38 (29.59 ×)	622.19×
$\text{ARCHER}_{\text{COP}}$	5110p co-processor	60 P, 4 T/P	3.33 (3.37 ×)	70.87×
$\text{ARCHER}_{\text{CPU+COP}}$	X5650 CPU and 5110p co-processor	CPU: 12 P, 1 T/P, coprocessor: 60 P, 4 T/P	2.59 (4.34 ×)	91.25×

To evaluate the performance of different hardware architectures, the parallel $\text{ARCHER}_{\text{CPU}}$ code is used as the basis to derive the speedup factors $\overline{\eta_P}$. Both

$\text{ARCHER}_{\text{GPU}}$ and $\text{ARCHER}_{\text{COP}}$ exhibit a good capability in accelerating MC calculations compared to the CPU counterpart running with 12 threads. In addition, $\text{ARCHER}_{\text{GPU}}$ was found to be faster than $\text{ARCHER}_{\text{COP}}$ by a factor of $60\% \sim 223\%$. It is also interesting to observe that in the case of 1 CPU and 6 M2090 GPUs, there is more number of batch iterations due to smaller number of streams used, and that the benefit of using CPU is suppressed by the increased period of low GPU occupancy, causing a slight performance reduction.

Another important performance factor worthy of comparison is the Figure Of Merit (FOM) (X-5 Monte Carlo Team 2003*a*), defined as the inverse of R^2T, where R is the relative standard deviation and T is the execution time. It is known that MCNPX adopts the more statistically efficient path-length estimator to calculate the radiation dose (X-5 Monte Carlo Team 2003*a*), which results in smaller R than the collision estimator used by ARCHER. However, it is found that the high computing efficiency of ARCHER significantly reduces T, offsets its slightly larger R and therefore enhances the overall FOM. We normalize the average FOMs across all the tallied doses by different codes to MCNPX's average FOM. The results are listed in table 3.5.

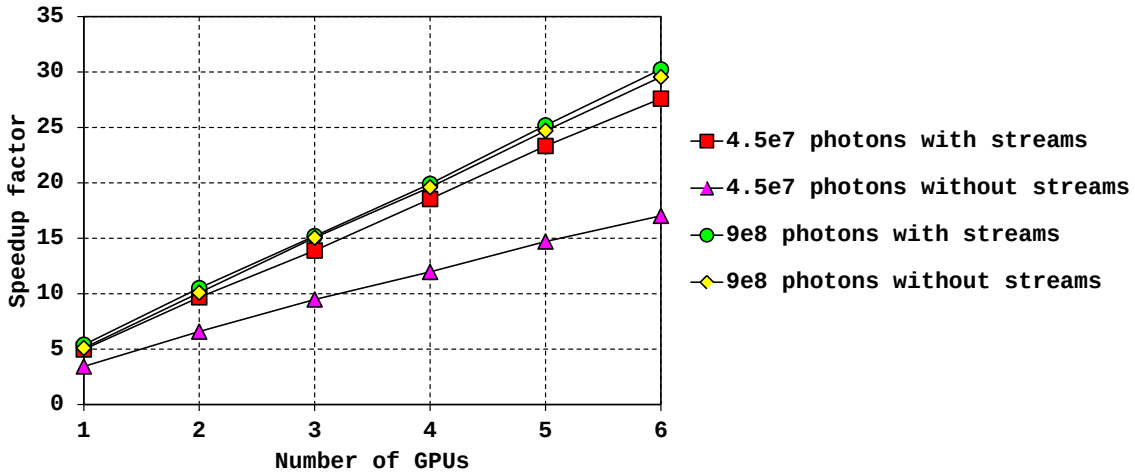


Figure 3.3: Performance of $\text{ARCHER}_{\text{GPU}}$ in a strong scaling problem using 1∼6 Nvidia M2090 GPUs. The computing task was the simulation of a whole-body axial scan over the RPI-Ault Male phantom (90 batches). The different total number of photons across all the batches are listed on the figure.

The scalability of $\text{ARCHER}_{\text{GPU}}$ is studied through a strong scaling problem, in which given a fixed-size Monte Carlo simulation, the computing efficiency is measured as the amount of hardware resource changed. The result in figure 3.3 shows that the speedup factors of $\text{ARCHER}_{\text{GPU}}$ over $\text{ARCHER}_{\text{CPU}}$ increases almost proportionally as the number of GPUs increased from 1 to 6, indicating a good scalability. The result also underscores the fact that the GPU stream implementation is capable of improving the performance when the number of photons in a single batch is not sufficiently large to consume the hardware resource of a single GPU. Specifically, for the M2090 GPU, the achieved occupancy (the actual number of active warps per SM divided by its maximum value) p is approximately 1/3. The Fermi-generation GPU allows a maximum of $w = 48$ active warps per SM and has a total of $s = 16$ SMs. Each warp contains $t = 32$ threads, and each thread simulates $n = 100$ photons. It follows that at least $N = pwstn = 819,200$ photons are required to fully occupy the GPU at runtime. When the computing task was reduced from 9×10^8 to 4.5×10^7 photons for 90 batches, the number of photons per batch is reduced from 1×10^7 ($> N$) to 5×10^5 ($< N$), leading to a hardware underutilization. This is effectively avoided by concurrently simulating 15 batches through GPU streams (Liu et al. 2014*a*).

3.2.1.2 Performance Comparison with Contemporary Study

The performance comparison between $\text{ARCHER}_{\text{GPU}}$ and Chen et al. (2012)'s GPU code is shown in table 3.6. The geometry of our abdomen phantom is altered to match with Chen et al. (2012)'s parameters. While $\text{ARCHER}_{\text{GPU}}$ underperforms by 16% in the low resolution case, it is much faster in the other two cases, and the performance improvement is more remarkable as the phantom resolution becomes higher (Liu et al. 2014*a*).

3.2.2 Energy Efficiency

The power draw by different computing devices as a function of time is plotted in figure 3.4. There are two qualifications regarding these curves. First, they apply to the parallel Monte Carlo photon transport process, and do not account for any other sequential code. Second, for $\text{ARCHER}_{\text{GPU}}$ and $\text{ARCHER}_{\text{COP}}$, the curves

Table 3.6: Performance comparison with Chen et al. (2012). The same geometric parameters and number of photons are chosen to be consistent with their study.

voxel resolution	voxel dimension $[mm^3]$	voxel number	total number of photons simulated	computation time by Chen et al. (2012) [sec]	computation time by $\text{ARCHER}_{\text{GPU}}$ [sec]	speedup
high	0.068× 0.068× 0.3	512× 512× 12	1.8×10^9	328.2	102.22	3.21×
medium	0.136× 0.136× 0.3	256× 256× 12	1.0×10^9	76.2	56.79	1.34×
low	0.272× 0.272× 0.3	128× 128× 12	0.5×10^9	24	28.63	0.84×

only consider the power consumed by the hardware accelerators and do not include that by the idle CPU. Clearly, the K20 GPU (Kepler) requires the least amount of instantaneous power supply, whereas the 5110p coprocessor demands the most compared to other computing devices.

The energy consumption, i.e. the power integrated over time, is listed in table 3.7. There are two sets of data, one ignoring the energy contribution from the idle CPU (labelled as device), the other including it (labelled as total). The contribution from the idle CPU is not significant and the rankings of the energy consumption by a certain computing device in these two sets of data are the same. Obviously, although the hardware accelerators in general requires higher power supply, they demonstrate excellent overall energy-saving capability. Particularly outstanding is the GPU platform. As the hardware evolves from M2090 GF110 chip, to K20 GK110 chip and to K40 GK110b chip, the energy budget for the given computing task is able to be significantly reduced. Further, when the GPU boost function is enabled on the K40 GPU, the benefit of higher GPU core frequency and faster code outweighs the cost of increased power usage. The coprocessor appears less energy efficient, surprisingly

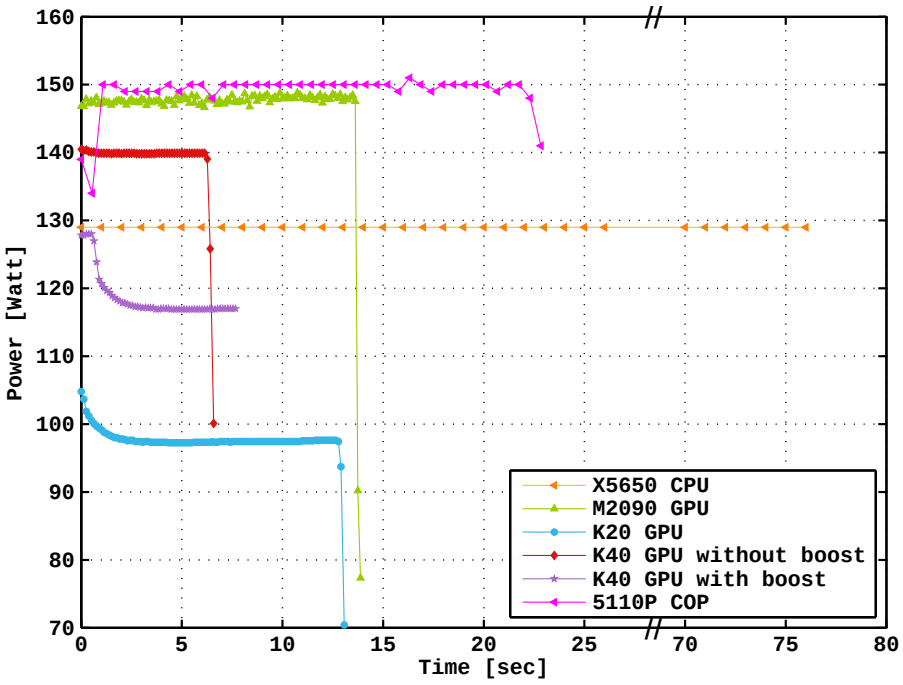


Figure 3.4: Comparison of the power draw by ARCHER variants run on the CPU, GPU and coprocessor platform respectively.

than the Fermi GPU that was released one and half year earlier. An alternative ground on which to compare the energy efficiency is the metric FLOPS per Watt, as is described in section 2.8.2. The result is shown in figure 3.5. This metric — important in the High Performance Computing (HPC) field — is interchangeable with the device energy consumption, because it is legitimate to assume that the FLOPS are identical across all the ARCHER variants (Liu et al. 2014*a*).

Table 3.7: Power and energy use by different codes on different hardware platforms. The computing task is the simulation of a single axial scan over the RPI-Adult Male 73kg phantom in the abdomen region (1 single batch with 1×10^8 photons). For the GPU and coprocessor cases, the power draw by the idle host is non-zero, and is included in the total energy consumption estimate.

code	hardware	device		total	
		average power draw [Watt]	energy consumption [Joule]	average power draw [Watt]	energy consumption [Joule]
$\text{ARCHER}_{\text{CPU}}$	X5650 CPU	129	9675	129	9675 (baseline)
$\text{ARCHER}_{\text{GPU}}$	M2090 GPU	137.04	2037.81	177.04	2632.61 (3.68×)
$\text{ARCHER}_{\text{GPU}}$	K20 GPU	98.69	1274.72	138.69	1791.37 (5.40×)
$\text{ARCHER}_{\text{GPU}}$	K40 GPU without boost	103.84	914.88	143.84	1267.29 (7.63×)
$\text{ARCHER}_{\text{GPU}}$	K40 GPU with boost	121.93	909.28	161.93	1207.59 (8.01×)
$\text{ARCHER}_{\text{COP}}$	5110p coprocessor	149.78	3406.03	189.78	4315.65 (2.24×)

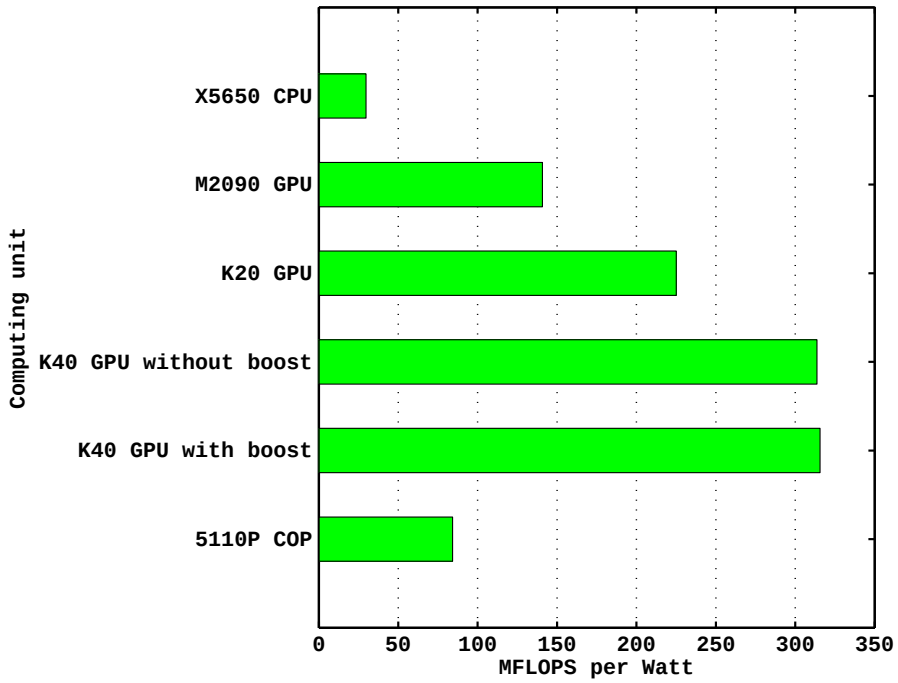


Figure 3.5: MFLOPS per Watt $\eta_{\mathbf{E}}$ (larger is better). As the time unit cancels out, this metric reflects for a given energy budget, the amount of work the hardware platform is able to do with the platform-specific code. For the hardware accelerators, $\eta_{\mathbf{E}}$ only accounts for the device energy usage. $\eta_{\mathbf{E}}$ is an equivalent quantity to the "device energy consumption" in table 3.7.

3.2.3 Cost Effectiveness

The parameters necessary for cost effectiveness analysis are listed in table 3.8. The price estimates were obtained on 06/22/2014 from various sources: the electricity rate is from US Energy Information Administration. (2014); the CPU price is from the online retailer Newegg. (2014*a*); the M2090 GPU price is from the online retailer Amazon. (2014*b*); the K20 GPU price is from Amazon. (2014*a*); the K40 GPU price is from Newegg. (2014*b*); the 5110p coprocessor price is from the server retailer Acmemicro. (2014). The calculated normalized cost effectiveness factors of different computing unit relative to the Intel Xeon 5650 CPU are plotted in figure 3.6. The initial values of $\overline{\eta_C^d(t)}$ can be used as a rough measure of the "worthiness" of a given computing unit. Suppose that, compared to a CPU, the hardware accelerator were twice faster, but four times more expensive, then $\overline{\eta_C^d(0)} = 0.5 < 1$, which means

without considering its long-term energy-saving capability, it would not be worthwhile to purchase the hardware accelerator as its performance was overshadowed by its price. Fortunately, this is not the case. On figure 3.6 all the initial values exceed 1, demonstrating a clear edge of the hardware accelerators over the CPU, and this advantage tends to increase over the time.

Table 3.8: Parameters requisite for cost effectiveness evaluation. The K40 GPU works in the boost mode.

parameter	meaning	value
κ	electricity rate per kWh	\$ 0.2087
$C_u^{CA,c}(0)$	capital cost of one Xeon 5650 CPU	\$ 799.99
$C_u^{CA,d}(0)$	capital cost of one M2090 GPU	\$ 1570
	capital cost of one K20 GPU	\$ 2695
	capital cost of one K40 GPU	\$ 5299.99
	capital cost of one 5110p coprocessor	\$ 2162
$\overline{P_c}$	average power draw by one Xeon 5650 CPU	129 W
$\overline{P_d}$	average power draw by one M2090 GPU	177.04 W
	average power draw by one K20 GPU	138.69 W
	average power draw by one K40 GPU	161.93 W
	average power draw by one 5110p coprocessor	189.78 W
η_P	speedup factor of M2090 GPU	5.40×
	speedup factor of K20 GPU	6.40×
	speedup factor of K40 GPU	10.89×
	speedup factor of 5110p coprocessor	3.37×

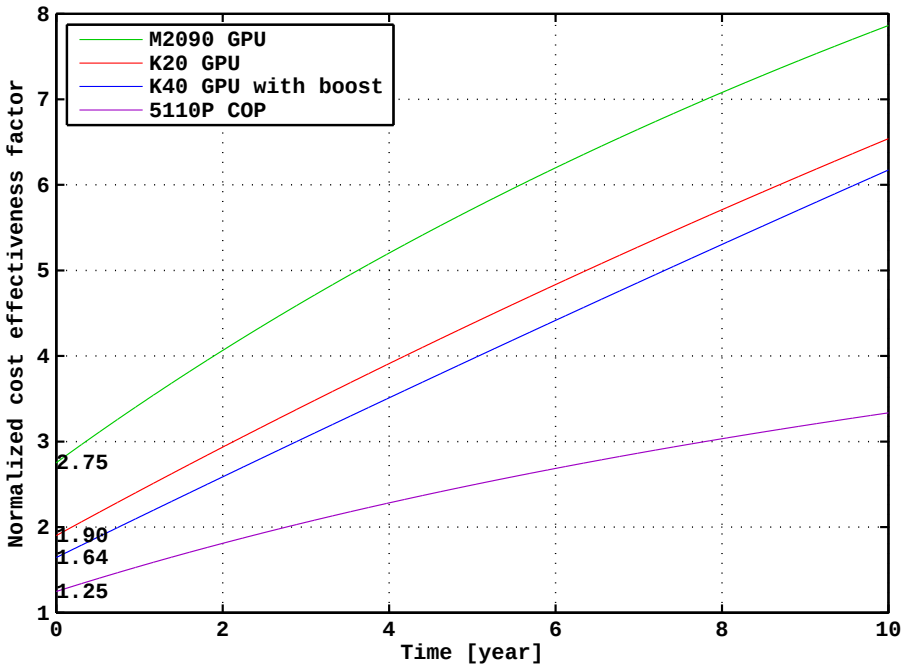


Figure 3.6: Normalized cost effectiveness factor $\overline{\eta_C^d(t)}$ (larger is better). The initial values ($\overline{\eta_C^d(0)}$) are labelled on the figure.

3.2.4 Profiling

The detailed profiling results of $\text{ARCHER}_{\text{GPU}}$ are shown in table 3.9, figure 3.7 and table 3.10. There are several interesting findings. First, although the code contains many conditional branches, the majority of them are not divergent. In other words, threads in a warp uniformly enter the same branch for most of the time. However, those divergent threads in a warp tend to execute a large amount of instructions to the extent that their peers are made inactive by the GPU mask for most of the time, resulting in a very low warp execution efficiency. This means that the "branch divergence" problem inherent in the GPU-based Monte Carlo code indeed reduces the hardware utilization. Second, the execution dependency appears to be the dominant reason for instruction stalls in figure 3.7. This mainly results from the very frequent access to the global and local memories, which are known to have long latency Nvidia. (2014). Third, for the global memory read in particular, the access pattern is scattered, irregular due to the random nature of Monte Carlo methods, and the GPU-favored coalesced memory access can hardly be achieved.

Consequently, a single memory load request from a warp typically leads to multiple actual transactions.

Table 3.9: Instruction statistics. The K40 GPU with compute capability of 3.5 does not provide a counter for branch efficiency evaluation. This specific quantity is measured on the M2090 GPU (Nvidia. 2013*g*).

metrics	meaning	value
issue slot utilization	Percentage of issue slots that issued at least one instruction, averaged across all cycles	47.93%
issued IPC	Instructions issued per cycle	2.35
executed IPC	Instructions executed per cycle	2.05
achieved occupancy	Ratio of the average active warps per active cycle to the maximum number of warps supported on a multiprocessor	42.18%
multiprocessor activity	The percentage of time at least one warp is active on a multiprocessor averaged over all multiprocessors on the GPU	95.93%
warp execution efficiency	Ratio of the average active threads per warp to the maximum number of threads per warp supported on a multiprocessor expressed as percentage	17.82%
warp non-predicated execution efficiency	Ratio of the average active threads per warp executing non-predicated instructions to the maximum number of threads per warp supported on a multiprocessor expressed as percentage	16.67%
branch efficiency	Ratio of non-divergent branches to total branches expressed as percentage	92.20%

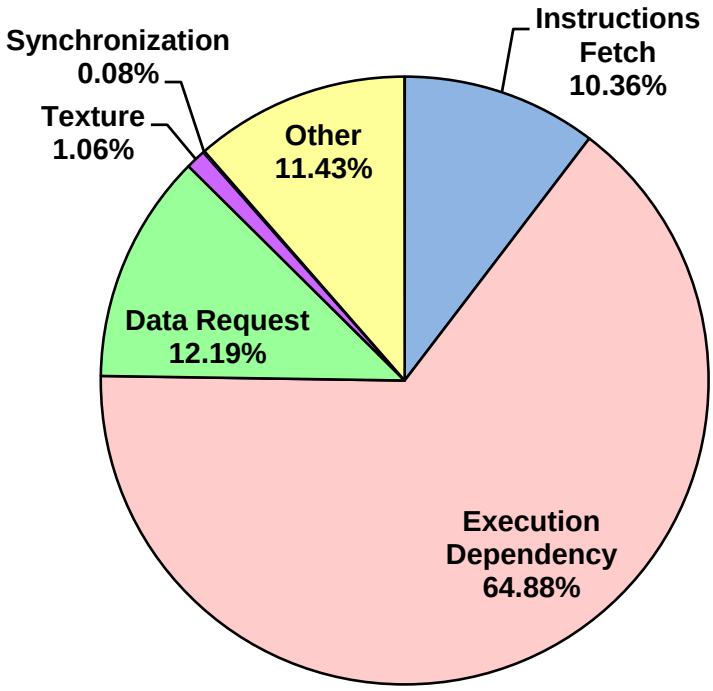


Figure 3.7: Statistics of instruction stall. The reasons for instruction stall can be put into 6 categories. (a)Instructions fetch — Percentage of stalls occurring because the next assembly instruction has not yet been fetched. (b) Execution dependency — percentage of stalls occurring because an input required by the instruction is not yet available. (c) Data request — percentage of stalls occurring because a memory operation cannot be performed due to the required resources not being available or fully utilized, or because too many requests of a given type are outstanding. (d) Texture — percentage of stalls occurring because the texture sub-system is fully utilized or has too many outstanding requests. (e) Synchronization — Percentage of stalls occurring because the warp is blocked at a `__syncthreads()` call. (f) Other — percentage of stalls occurring due to miscellaneous reasons (Nvidia. 2013*g*).

Table 3.10: Memory statistics (Nvidia. 2013*g*).

metrics	meaning	value
L1 global hit rate	Hit rate in L1 cache for global loads	86.44%
L1 local hit rate	Hit rate in L1 cache for local loads and stores	21.39%
L2 hit rate (L1 reads)	Hit rate at L2 cache for all read requests from L1 cache	50.27%
L2 hit rate (texture reads)	Hit rate at L2 cache for all read requests from texture cache	19.14%
texture cache hit rate	Texture cache hit rate	1.85%
global memory load efficiency	Ratio of requested global memory load throughput to required global memory load throughput expressed as percentage	6.30%
global memory store efficiency	Ratio of requested global memory store throughput to required global memory store throughput expressed as percentage	30.03%

3.3 Clinical Applications

In the clinical application test, $\text{ARCHER}_{\text{GPU}}$ is used to calculate 3-D dose distribution, i.e. the absorbed doses recorded in a voxel-by-voxel manner. Three cross-sections of the dose matrix are used to generate the dose map in figure 3.8. The statistical requirement is such that the doses to all the well-segmented organs have a relative standard deviation less than 1% (Liu et al. 2014*a*).

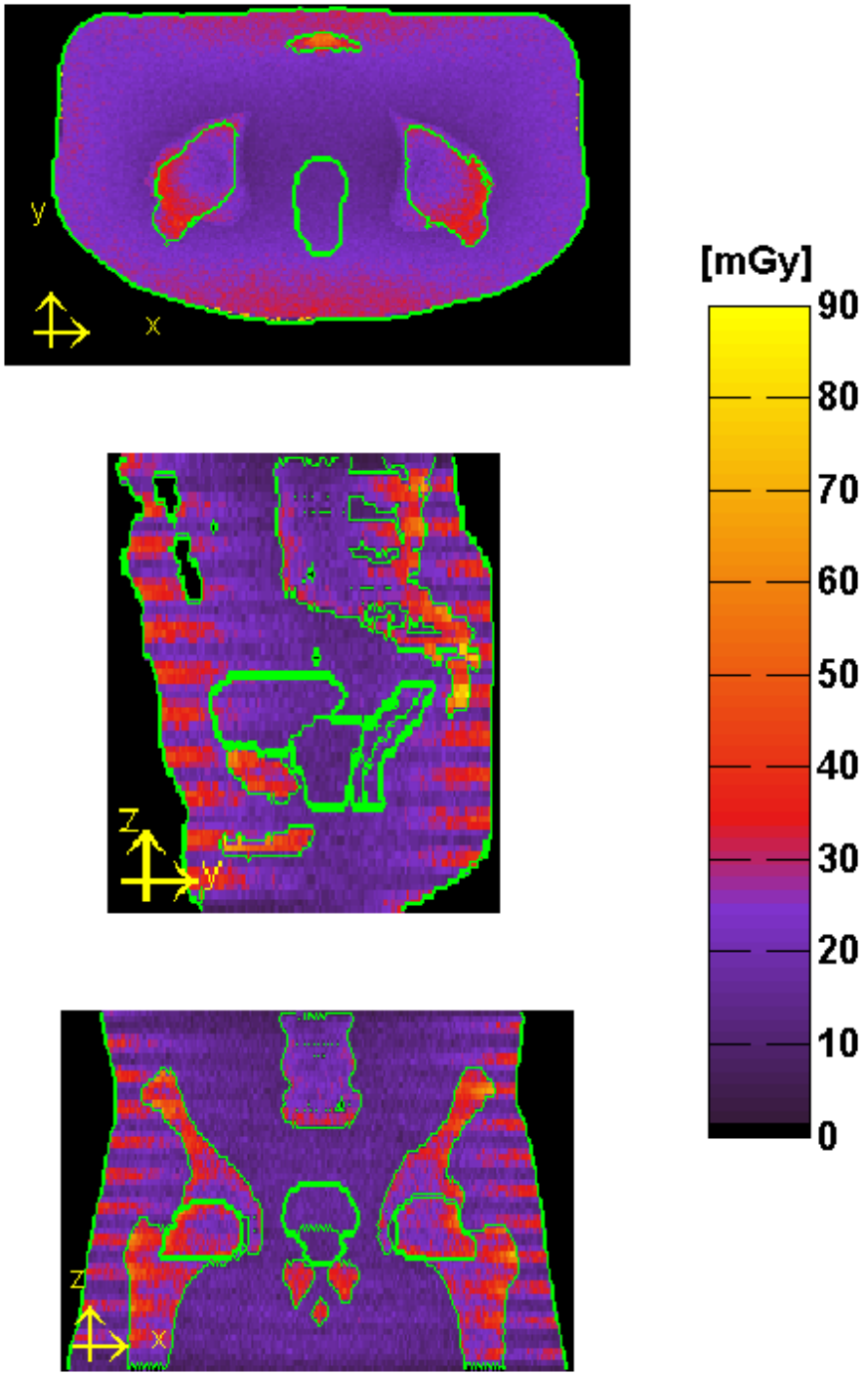


Figure 3.8: Calculated CT dose distributions with the prostate, rectum, urinary bladder and femoral head outlined in green.

The computing efficiency of $\text{ARCHER}_{\text{GPU}}$ on different GPU models is listed in table 3.11. This denotes that $\text{ARCHER}_{\text{GPU}}$ performs very fast in our specific test

case, and that the code scales very well when the number of hardware devices is increased from 1 to 6.

Table 3.11: Computing efficiency of ARCHER$_{\mathbf{GPU}}$ in the clinical 3-D dose distribution calculations.

hardware accelerator	execution time [sec]
1 M2090 GPU	6.74
6 M2090 GPUs	1.23
1 K20 GPU	5.60
1 K40 GPU with boost	3.42

3.4 Long-term Development of ARCHER for CT

According to Moore's law (Moore 1998) — which arguably still maintains its validity — the exascale computing era will arrive in around 2020. Along with it will be the drastic change in the hardware architectures and programming models. The hardware accelerators come into being and rapidly develop in this context. Distinguished in delivering high FLOPS and having high energy efficiency, they require developers to either use new platform-specific programming models or fine-tune the code with new platform-specific tools. On the other hand, the heated competition between hardware manufacturers in the High Performance Computing (HPC) industry makes multiple important questions unanswered — which platform may dominate in the future, which programming model is the best to select and stick to, whether it is worth the labor and time to port the existing code to the new architecture, or it is wiser to wait for the conventional hardware to evolve.

The new computing technologies will impact our nuclear engineering and medical physics communities, and expand our view from the computing performance alone to scalability, energy efficiency, environmental friendliness and a lot more. With that in mind, we have initiated this long-term research program ARCHER to develop a new generation of parallel, scalable Monte Carlo package targeted for different computing platforms (figure 3.9). ARCHER is envisioned as a versatile test

bed to explore how the new parallel computing platforms can potentially benefit different applications in our field, to which extent and at what cost (Xu et al. 2013, Liu et al. 2014*a*). This study specifically focuses on the photon transport for the CT imaging dosimetry application. Other studies we have been conducting include the electron transport for the radiotherapy application (Su et al. 2014) and neutron transport for the reactor analysis application (Ding et al. 2011, Liu et al. 2012).

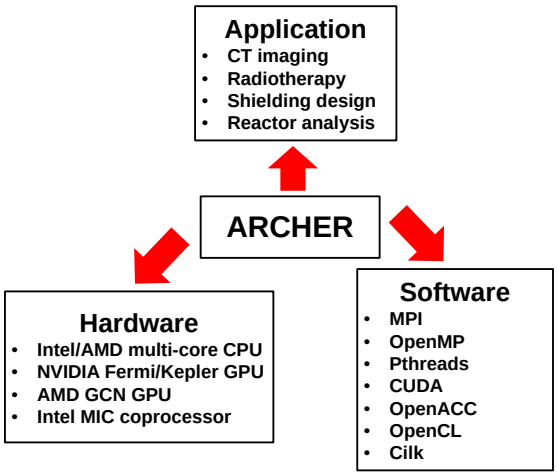


Figure 3.9: ARCHER is envisioned as a versatile test bed for the modern and future parallel computing platforms. It will be designed as a package of application-oriented codes, having several versions written in different programming languages, and optimized to different platforms.

CHAPTER 4
CONCLUSIONS

"If one has really technically penetrated a subject, things that previously seemed in complete contrast, might be purely mathematical transformations of each other."

—von Neumann, John

4.1 Summary

This research applied the emerging HPC technologies — the GPU and coprocessor — to the field of Monte Carlo radiation dosimetry. A new parallel Monte Carlo package named ARCHER was developed to enable fast and accurate patient-specific X-ray computed tomography (CT) dose calculations. ARCHER had three components, the parallel CPU code, the GPU code and the coprocessor code, that helped us evaluate the performance of various computing platforms.

Corresponding to our objectives, this research has the following conclusions.

- *ARCHER is a Monte Carlo simulation tool with practical value and good flexibility.* From users' perspective, ARCHER has three features. First, it provides convenience for dose estimates by incorporating a built-in model of the GE LightSpeed 16 Pro CT scanner and a library of preset heterogeneous phantoms. Second, it can convert the clinical CT data, i.e. the Digital Imaging and Communications in Medicine (DICOM) images into heterogeneous phantoms. Third, it is parallel, highly optimized, and can run on three types of hardware platforms, the multi-core CPU, the Nvidia GPU and the Intel coprocessor.

Portions of this chapter are to appear in: Liu, T., Du, X., Su, L., Ji, W., Carothers, C. D., Shephard, M. S., Liu, B., Kalra, M., Brown, F. B., Fitzgerald, P. F. and Xu, X. G. (2014), 'ARCHER-CT, an extremely fast Monte Carlo code for patient-specifc ct dose calculations using NVIDA GPU and Intel coprocessor technologies: part I — software development and testing', *Phys. Med. Biol.*. (submitted).

These three features make ARCHER a valuable Monte Carlo simulator in the radiation dosimetry research as well as in the clinical application. They also allow ARCHER to become a testbed to assess the practical value of different parallel platforms of today and tomorrow.

- *ARCHER has accurate transport physics model.* The benchmark test against the production code Monte Carlo N-Particle eXtended (MCNPX) showed that the photon transport model in ARCHER was very accurate, and that ARCHER could provide reliable dose estimates provided the input simulation models were accurate. The comparison with the experiment using the real human subject exhibited a dose discrepancy by a factor of 29%, indicating that there was room for improvement on the current CT scanner model and the phantom generation algorithm.

- *For CT dose calculations, both the GPU and coprocessor codes of ARCHER have higher computing, energy efficiency and cost effectiveness, and the GPU platform takes the large lead.* The performance tests demonstrated several facts. First, $\text{ARCHER}_{\text{GPU}}$ run on an M2090 (Fermi), K20 (Kepler) or K40 (Kepler) GPU had higher computing efficiency than $\text{ARCHER}_{\text{CPU}}$ on a 6-core Intel Xeon X5650 CPU (Westmere) by a factor of $5.40 \sim 10.89$, while $\text{ARCHER}_{\text{COP}}$ on an Intel Xeon Phi 5110p coprocessor (Knights Corner) was faster than the CPU code by a factor of 3.37. Second, compared to $\text{ARCHER}_{\text{CPU}}$, $\text{ARCHER}_{\text{GPU}}$ was more energy-efficient by a factor of $3.68 \sim 8.01$, while $\text{ARCHER}_{\text{COP}}$ was by a factor of 2.24. Third, using the cost-effectiveness model we established, the GPU platform was found to be $1.64 \sim 2.75$ times more economical than the CPU, and the coprocessor platform was 1.25 times better than the CPU. Fourth, the GPU stream implementation allowed $\text{ARCHER}_{\text{GPU}}$ to maintain good scalability. Fifth, exploitation of system concurrency at different scales — including multiple GPU grids on a single GPU, multiple GPUs, and CPU-GPU or CPU-coprocessor — effectively improved the performance. Particularly, it was concluded that the GPU platform, from the past Fermi architecture to the current Kepler, outperformed the coprocessor platform of the current Knights

Corner architecture in terms of the computing efficiency and the energy efficiency.

- *ARCHER has clinical potentials.* In the clinical application, it was found that $\text{ARCHER}_{\text{GPU}}$ had good performance, taking 3.42 seconds on a K40 GPU and 1.23 seconds on 6 M2090 GPUs to finish the calculations of CT imaging dose distribution. The current paradigm of CT organ dose calculation is based on the "indirect and population-averaged patient phantom" approach and carried out in the off-line retrospective studies. This result suggests that the new paradigm of "direct and patient-specific" CT dose calculation is feasible with the use of an efficient Monte Carlo computing engine as an integral part of the CT imaging process. Doubtless, such a capability will facilitate new research in CT image quality optimization and dose management.

4.2 Future Work

Future work will focus on addressing several limitations in this research (Liu et al. 2014*a*) and extending the functions of ARCHER. The specific tasks are listed below.

- In the dose distribution calculations, for simplicity, we directly used the number of particles that satisfied the statistical requirement for the case of organ dose calculation. Ideally there should be a dynamic approach to run the dose distribution calculation in batches using a preset number of particles, check the statistics between batches at runtime, and terminate the calculation once certain statistical requirements imposed on the ROI are met.

- The energy analysis adopted the sophisticated software approach and only applied to the processing units, and not to the system memory, hard drive and cooling device, whereas a more straightforward and thorough method will be using the external power meter to obtain the electricity-related information from the heterogeneous computing system.

- Successful realization of accurate, fast, patient-specific organ dose calculation in clinics requires a chain of efficient tools. ARCHER constitutes the last step

of computing and reporting radiation doses provided that the accurate patient anatomical data are readily available. The tools that precede ARCHER will be the ones capable of performing fast and reliable image reconstruction, segmentation or outline, and construction of the patient phantoms. These tools should be carefully surveyed in the future.

- The electron module of ARCHER (Su et al. 2014) currently adopts a simplistic photon transport treatment. Future work is to replace this part with the accurate transport modelling developed in this study.

- The visualization module is currently being developed to display the CT scanner, computational phantom and/or the photon trajectories in 3-D rendered images. Two approaches are being used. One simple approach is to record the photon spatial information, save it as a Non-uniform rational basis spline (NURBS) object to a Wavefront .obj file (Wavefront Technologies. 2013), and later visualize it in commercial 3-D modelling software such as Rhinoceros (Robert McNeel & Associates. 2014). The other advanced approach is to visualize all the information by OpenGL (Woo et al. 1999) and give users full freedom to customize the image rendering.

- A generic front end is being developed to achieve the global level of concurrency, i.e. the concurrent execution of the CPU, GPU and coprocessor together. This will maximize the hardware usage and further improve the computing efficiency of ARCHER on our heterogeneous computing system. The front end is written in OpenCL (Khronos OpenCL Working Group. 2008) and will be able to detect the number and type of computing units, split the simulation task properly, and invoke the existing CPU and hardware accelerator codes.

REFERENCES

Acmemicro. (2014), Intel Xeon Phi coprocessor 5110P price. http://www.acmemicro.com/Product/11972/Intel-Xeon-Phi-Coprocessor-5110P-8GB-1-053-GHz-60-core-x16-PCI-E-2-0-Passive?gclid=CjgKEAjwt4-dBRCDnaTUn-mC_0oSJAC4Q6kG8fI3u5z-kHR0QlkL8ImkG_m4un1w1MBhCbwFOrqfR_D_BwE (Retrieved on July 09, 2014).

Agostinelli, S., Allison, J., Amako, K. e., Apostolakis, J., Araujo, H., Arce, P., Asai, M., Axen, D., Banerjee, S. & Barrand, G. (2003), 'GEANT4–a simulation toolkit', *Nucl. Instrum. Meth. A* **506**(3), 250–303.

Amazon. (2014*a*), Nvidia Tesla K20 GPU price. http://www.amazon.com/NVIDIA-Tesla-K20-Accelerator-900-22081-2220-000/dp/B00AA2C1DC (Retrieved on July 09, 2014).

Amazon. (2014*b*), Nvidia Tesla M2090 GPU price. http://www.amazon.com/Nvidia-Tesla-M2090-Gpu-Card/dp/B005TJKPWU (Retrieved on July 09, 2014).

AMD. (2012), APU 101: All about AMD Fusion accelerated processing units, and how they're changing, well, everything. http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/apu101.pdf (Retrieved on July 09, 2014).

Amis Jr, E. S., Butler, P. F., Applegate, K. E., Birnbaum, S. B., Brateman, L. F., Hevezi, J. M., Mettler, F. A., Morin, R. L., Pentecost, M. J. & Smith, G. G. (2007), 'American College of Radiology white paper on radiation dose in medicine', *J. Am. Coll. Radiol.* **4**(5), 272–284.

Arnow, B. J. (1994), 'On Laplace's extension of the Buffon needle problem', *Coll. Math. J.* **25**(1), 40–43.

Attix, F. H. (2008), *Introduction to radiological physics and radiation dosimetry*, John Wiley & Sons, Weinheim, Germany.

Badal, A. & Badano, A. (2009), 'Accelerating Monte Carlo simulations of photon transport in a voxelized geometry using a massively parallel graphics processing unit', *Med. Phys.* **36**(11), 4878–4880.

Badal, A. & Badano, A. (2011), 'Fast and accurate estimation of organ doses in medical imaging using a GPU-accelerated Monte Carlo simulation', *Med. Phys.* **38**(6), 3411.

Baró, J., Sempau, J., Fernández-Varea, J. & Salvat, F. (1995), 'PENELOPE: An algorithm for Monte Carlo simulation of the penetration and energy loss of electrons and positrons in matter', *Nucl. Instrum. Meth. B* **100**(1), 31–46.

Bayoumi, T., Reda, S. & Saleh, H. (2012), 'Assessment study for multi-barrier system used in radioactive borate waste isolation based on Monte Carlo simulations', *Appl. Radiat. Isot.* **70**(1), 99–102.

Berners-Lee, T. & Connolly, D. (1995), Hypertext markup language-2.0, Report RFC 1866, MIT/W3C.

Berrington de González, A. & Darby, S. (2004), 'Risk of cancer from diagnostic X-rays: Estimates for the UK and 14 other countries', *The Lancet* **363**, 345–351.

Bobrowicz, F. W., Lynch, J. E., Fisher, K. J. & Tabor, J. E. (1984), 'Vectorized Monte Carlo photon transport', *Parallel. Comput.* **1**(3), 295–305.

Boone, J. M., Strauss, K. J., Cody, D. D., McCollough, C. H., McNitt-Gray, M. F., Toth, T. L., Goske, M. J. & Frush, D. P. (2011), Size-specific dose estimates (SSDE) in pediatric and adult body CT examinations, Report AAPM No. 204, American Association of Physicists in Medicine (AAPM).

Brenner, D. & Huda, W. (2008), 'Effective dose: A useful concept in diagnostic radiology?', *Radiat. Prot. Dosimetry* **128**(4), 503–508.

Brenner, D. J. (2002), 'Estimating cancer risks from pediatric CT: Going from the qualitative to the quantitative', *Pediatr. Radiol.* **32**(4), 228–231.

Brenner, D. J. & Hall, E. J. (2007), 'Computed Tomography-An Increasing Source of Radiation Exposure', *N. Engl. J. Med.* **357**(22), 2277–2284.

Brown, F. B. (2005), Fundamentals of Monte Carlo particle transport, Report LA-UR-05-4983, Los Alamos National Laboratory (LANL).

Brown, F. B. (2011), 'Recent advances and future prospects for Monte Carlo', *Prog. Nucl. Sci. Technol.* **2**, 1–4.

Brown, F. B. & Martin, W. R. (1984), 'Monte Carlo methods for radiation transport analysis on vector computers', *Prog. Nucl. Energ.* **14**(3), 269–299.

Buffon, G. (1777), 'Essai d'arithmétique morale', *Histoire naturelle, générale er particulière, Supplément* **4**, 46–123.

Carrier, J. F., Archambault, L., Beaulieu, L. & Roy, R. (2004), 'Validation of GEANT4, an object-oriented Monte Carlo toolkit, for simulations in med. phys.', *Med. Phys.* **31**(3), 484–492.

Cashwell, E. D., Neergaard, J. R., Everett, C. J., Schrandt, R. G., Taylor, W. M. & Turner, G. D. (1973), Monte Carlo Photon Codes: MCG and MCP, Report LA-5157-MS, Los Alamos National Laboratory (LANL).

Chacon, S. (2009), Pro Git. `http://git-scm.com/docs` (Retrieved on July 09, 2014).

Chen, W., Kolditz, D., Beister, M., Bohle, R. & Kalender, W. A. (2012), 'Fast on-site Monte Carlo tool for dose calculations in CT applications', *Med. Phys.* **39**(6), 2985–2996.

Childress, N. L. & Miller, W. H. (2002), 'MCNP analysis and optimization of a triple crystal phoswich detector', *Nucl. Instrum. Meth. A* **490**(1), 263–270.

Şeker, V. & Çolak, U. (2003), 'HTR-10 full core first criticality analysis with MCNP', *Nucl. Eng. Des.* **222**(2), 263–270.

Dally, B. (2010), GPU computing to exascale and beyond, *in* 'The International Conference for High Performance Computing, Networking, Storage and Analysis (SC10)', New Orleans, LA.

Ding, A. (2012), Development of a radiation dose reporting software for X-ray Computed Tomography (CT), Thesis, Rensselaer Polytechnic Institute (RPI), Department of Mechanical, Aerospace, and Nuclear Engineering.

Ding, A., Gu, J., Trofimov, A. V. & Xu, X. G. (2010), 'Monte Carlo calculation of imaging doses from diagnostic multidetector CT and kilovoltage cone-beam CT as part of prostate cancer treatment plans', *Med. Phys.* **37**(12), 6199–6204.

Ding, A., Liu, T., Liang, C., Ji, W., Shepard, M. S., Xu, X. G. & Brown, F. B. (2011), Evaluation of speedup of Monte Carlo calculations of simple reactor physics problems coded for the GPU/CUDA environment, *in* 'Proceedings of International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)', American Nuclear Society (ANS), Rio de Janeiro, Brazil.

Ding, A., Mille, M., Liu, T., Caracappa, P. F. & Xu, X. G. (2012*b*), '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**(9), 2441–2459.

Dongarra, J. (2013), Performance Application Programming Interface (PAPI) 5.3.0.0 reference. `http://icl.cs.utk.edu/papi/docs/` (Retrieved on July 09, 2014).

El-Guebaly, L. (1997), 'Overview of ARIES-RS neutronics and radiation shielding: Key issues and main conclusions', *Fusion. Eng. Des.* **38**(1), 139–158.

Everett, C. & Cashwell, E. (1973), MCP code fluorescence-routine revision, Report LA-5240-MS, Los Alamos National Laboratory (LANL).

Ferrari, A., Sala, P. R., Fasso, A. & Ranft, J. (2005), Fluka: A multi-particle transport code, Report SLAC-R-773, Conseil Européen pour la Recherche Nucléaire (CERN).

Gao, Y., Ding, A., Caracappa, P. F., Xu, X. G., Zhang, D. & Liu, B. (2013), Tube current modulated Computed Tomography simulation and dose calculation with Monte Carlo method, *in* 'American Nuclear Society 2013 Student Conference', Boston, MA.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. & Sunderam, V. (1994), *PVM: Parallel virtual machine: A users' guide and tutorial for networked parallel computing*, Scientific and engineering computation series, MIT Press, Boston, MA.

Goske, M. J., Applegate, K. E., Boylan, J., Butler, P. F., Callahan, M. J., Coley, B. D., Farley, S., Frush, D. P., Hernanz-Schulman, M., Jaramillo, D., Johnson, N. D., Kaste, S. C., Morrison, G., Strauss, K. J. & Tuggle, N. (2008), 'The Image Gently campaign: Working together to change practice', *Am. J. Roentgenol.* **190**(2), 273–274.

Green500. (2013), Green 500 supercomputer sites. `http://www.green500.org/greenlists` (Retrieved on July 09, 2014).

Gropp, W. (2002), 'MPICH2: A new start for MPI implementations', *Lect. Notes Comput. Sc.* **2474**, 7.

Gu, J. (2010), Development of CT scanner models for patient organ dose calculations using Monte Carlo methods, Thesis, Rensselaer Polytechnic Institute (RPI), Department of Mechanical, Aerospace, and Nuclear Engineering.

Gu, J., Bednarz, B., Caracappa, P. F. & Xu, X. G. (2009), '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**(9), 2699–2717.

Harris, M. (2011), Optimizing parallel reduction in CUDA. `http://developer.download.nvidia.com/assets/cuda/files/reduction.pdf` (Retrieved on July 09, 2014).

Health Physics Society. (2010), Fact Sheet: Radiation exposure from medical exams and procedures. `http://hps.org/documents/Medical_Exposures_Fact_Sheet.pdf` (Retrieved on July 09, 2014).

Hennessy, J. L. & Patterson, D. A. (2012), *Computer architecture: A quantitative approach*, five edn, Elsevier, Waltham, MA.

Hissoiny, S., Ozell, B., Bouchard, H. & Després, P. (2011), 'GPUMCD: A new GPU-oriented Monte Carlo dose calculation platform', *Med. Phys.* **38**(2), 754–764.

ICRP (2007), The 2007 recommendations of the International Commission on Radiological Protection, Report ICRP 103, International Commission on Radiological Protection (ICRP).

Intel. (2003), Intel Hyper-Threading technology technical user's guide. http://cache-www.intel.com/cd/00/00/01/77/17705_htt_user_guide.pdf (Retrieved on July 09, 2014).

Intel. (2010), Intel Many Integrated Core architecture. http://www.many-core.group.cam.ac.uk/ukgpucc2/talks/Elgar.pdf (Retrieved on July 09, 2014).

Intel. (2012), Intel xeon phi coprocessor 5110p specifications. http://ark.intel.com/products/71992/Intel-Xeon-Phi-Coprocessor-5110P-8GB-1 (Retrieved on July 09, 2014).

Intel. (2013*a*), How to use huge pages to improve application performance on intel xeon phi coprocessor. https://software.intel.com/sites/default/files/Large_pages_mic_0.pdf (Retrieved on July 09, 2014).

Intel. (2013*b*), Intel C++ compiler XE 13.1 user and reference guide. https://software.intel.com/sites/products/documentation/doclib/stdxe/2013/composerxe/compiler/cpp-win/index.htm (Retrieved on July 09, 2014).

Intel. (2013*c*), Intel xeon phi coprocessor 3120p specifications. http://ark.intel.com/products/75798/Intel-Xeon-Phi-Coprocessor-3120P-6GB-1 (Retrieved on July 09, 2014).

Intel. (2013*d*), Intel xeon phi coprocessor 7120p specifications. http://ark.intel.com/products/80310/Intel-Xeon-Phi-Coprocessor-7120D-16GB-1 (Retrieved on July 09, 2014).

Intel. (2013*e*), Intel Xeon Phi coprocessor system software developers guide. https://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-system-software-developers-guide (Retrieved on July 09, 2014).

Intel. (2013*f*), Intel Xeon Phi product family performance. http://www.intel.com/content/dam/www/public/us/en/documents/performance-briefs/xeon-phi-product-family-performance-brief.pdf (Retrieved on July 09, 2014).

Intel. (2013*g*), White paper: System administration for the Intel Xeon Phi coprocessor. https://software.intel.com/sites/default/files/article/373934/system-administration-for-the-intel-xeon-phi-coprocessor.pdf (Retrieved on July 09, 2014).

Intel. (2014), Intel MPI library for Linux OS reference manual. https://prd1idz.cps.intel.com/sites/products/documentation/hpc/ics/impi/41/lin/Reference_Manual/Reference_Manual.pdf (Retrieved on July 09, 2014).

Jahnke, L., Fleckenstein, J., Wenz, F. & Hesser, J. (2012), 'GMC: A GPU implementation of a Monte Carlo dose calculation based on Geant4 Phys', *Phys. Med. Biol.* **57**(5), 1217–1229.

Jeffers, J. & Reinders, J. (2013), *Intel Xeon Phi Coprocessor High Performance Programming*, Newnes, Boston, MA.

Jia, X., Gu, X., Graves, Y. J., Folkerts, M. & Jiang, S. B. (2011), 'GPU-based fast Monte Carlo simulation for radiotherapy dose calculation', *Phys. Med. Biol.* **56**(22), 7017–7031.

Jia, X., Gu, X., Sempau, J., Choi, D., Majumdar, A. & Jiang, S. B. (2010), 'Development of a GPU-based Monte Carlo dose calculation code for coupled electron-photon transport', *Phys. Med. Biol.* **55**(11), 3077–3086.

Jia, X., Yan, H., Gu, X. & Jiang, S. B. (2012), 'Fast Monte Carlo simulation for patient-specific CT/CBCT imaging dose calculation', *Phys. Med. Biol.* **57**(3), 577–590.

Johnson, P. B., Whalen, S. R., Wayson, M., Juneja, B., Lee, C. & Bolch, W. E. (2009), 'Hybrid patient-dependent phantoms covering statistical distributions of body morphometry in the US adult and pediatric population', *Proc. IEEE* **97**(12), 2060–2075.

Kardjilov, N., De Beer, F., Hassanein, R., Lehmann, E. & Vontobel, P. (2005), 'Scattering corrections in neutron radiography using point scattered functions', *Nucl. Instrum. Meth. A* **542**(1), 336–341.

Kawrakow, I. & Rogers, D. (2000), The EGSnrc code system: Monte Carlo simulation of electron and photon transport, Technical report, National Research Council Canada.

Kelvin, L. (1901), 'Nineteenth century clouds over the dynamical theory of heat and light', *Lon. Edin. Dub. Phil. Magaz. J. Sci.* **2**(7), 1–40.

Khronos OpenCL Working Group. (2008), The OpenCL specification. `https://www.khronos.org/registry/cl/specs/opencl-1.0.29.pdf` (Retrieved on July, 09, 2014).

Kleijnen, J. P. (1995), 'Verification and validation of simulation models', *Eur. J. Oper. Res.* **82**(1), 145–162.

Kogge, P., Amarasinghe, S., Campbell, D., Carlson, W., Chien, A., Dally, W., Elnohazy, E., Hall, M., Harrison, R., Harrod, W., Hill, K., Hiller, J., Karp, S., Koelbel, C., Koester, D., Levesque, J., Reed, D., Sarkar, V., Schreiber, R., Richards, M., Scarpelli, A., Shalf, J., Snavely, A. & Sterling, T. (2008), Exascale computing

study: Technology challenges in achieving exascale systems, Technical report, Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO). `http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf` (Retrieved on July, 09, 2014).

Larson, D. B., Johnson, L. W., Schnell, B. M., Salisbury, S. R. & Forman., H. P. (2011), 'National trends in CT use in the emergency department: 1995-2007', *Radiology* **258**(1), 164–173.

Lee, V. W., Kim, C., Chhugani, J., Deisher, M., Kim, D., Nguyen, A. D., Satish, N., Smelyanskiy, M., Chennupaty, S., Hammarlund, P., Singhal, R. & Dubey, P. (2010), 'Debunking the 100X GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU', *ACM SIGARCH Comp. Arch. News* **38**(3), 451–460.

Lewis, E. E. & Miller, Jr., W. F. (1984), *Computational methods of neutron transport*, John Wiley & Sons, Weinheim, Germany.

Lewis, M. (2005), ImPACT technology update No. 3: Radiation dose issues in multi-slice CT scanning, Technical report, St. George's Hospital. `http://www.impactscan.org/` (Retrieved on July 09, 2014).

Li, X., Samei, E Segars, W. P., Sturgeon, G. M., Colsher, J. G., Toncheva, G., Yoshizumi, T. T. & Frush, D. P. (2010), 'Patient-specific radiation dose and cancer risk estimation in CT: Part I. Development and validation of a Monte Carlo program', *Med. Phys.* **38**(1), 397–407.

Li, X., Samei, E Segars, W. P., Sturgeon, G. M., Colsher, J. G., Toncheva, G., Yoshizumi, T. T. & Frush, D. P. (2011), 'Patient-specific radiation dose and cancer risk estimation in CT: Part II. Application to patients', *Med. Phys.* **38**(1), 408–419.

Lindholm, E., Nickolls, J., Oberman, S. & Montrym, J. (2008), 'NVIDIA Tesla: A unified graphics and computing architecture', *IEEE Micro* **28**(2), 39–55.

Liu, T., Ding, A., Caracappa, P. F. & Xu, X. G. (2011), Modeling of obese individuals using automatic deformation of mesh-based computational phantoms, *in* '56th Annual Meeting of the Health Physics Society', West Palm Beach, FL.

Liu, T., Ding, A., Ji, W., Xu, X. G., Carothers, C. D. & Brown, F. B. (2012), A Monte Carlo neutron transport code for eigenvalue calculations on a dual-GPU system and CUDA environment, *in* 'International Topical Meeting on Advances in Reactor Physics (PHYSOR 2012)', American Nuclear Society (ANS), Knoxville, TN.

Liu, T., Ding, A. & Xu, X. G. (2012*a*), 'Accelerated Monte Carlo methods for photon dosimetry using a dual-GPU system and CUDA', *Med. Phys.* **39**(6), 3818.

Liu, T., Du, X., Su, L., Gao, Y., Ji, W., Zhang, D., Shi, J. Q., Liu, B., Kalra, M. K. & Xu, X. G. (2014), Monte Carlo CT dose calculation: A comparison between experiment and simulation using ARCHER-CT, *in* 'AAPM 56th Annual Meeting & Exhibition', Austin, TX.

Liu, T., Du, X., Su, L., Ji, W., Carothers, C. D., Shephard, M. S., Liu, B., Kalra, M., Brown, F. B., Fitzgerald, P. F. & Xu, X. G. (2014*a*), 'ARCHER-CT, an extremely fast Monte Carlo code for patient-specific ct dose calculations using NVIDA GPU and Intel coprocessor technologies: Part I — software development and testing', *Phys. Med. Biol.* . (submitted).

Liu, T., Ji, W. & Xu, X. G. (2013), Development of GPU-based Monte Carlo code for fast CT imaging dose calculation on CUDA Fermi architecture, *in* 'International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)', American Nuclear Society (ANS), Sun Valley, ID, pp. 1199–1210.

Love, E., Pauley, K. & Reid, B. (1995), Use of MCNP for characterization of reactor vessel internals waste from decommissioned nuclear reactors, Report INEL-95/0419, Idaho National Laboratory (INL).

Mack, C. A. (2011), 'Fifty years of Moore's law', *IEEE Trans. Semicond. Manuf.* **24**(2), 202–207.

Marsaglia, G. (2003), 'Xorshift RNGs', *J. Stat. Softw.* **8**(14), 1–6.

Mathews, J. D., Forsythe, A. V., Brady, Z., Butler, M. W., Goergen, S. K., Byrnes, G. B., Giles, G. G., Wallace, A. B., Anderson, P. R., Guiver, T. A., McGale, P., Cain, T. M., Dowty, J. G., Bickerstaffe, A. C. & Darby, S. C. (2013), 'Cancer risk in 680,000 people exposed to computed tomography scans in childhood or adolescence: Data linkage study of 11 million Australians', *BMJ* **346**, 1–18.

MathWorks. (1996), Matlab language reference manual. `http://www.mathworks.com/help/matlab/` (Retrieved on July 09, 2014).

McCollough, C., Cody, D., Edyvean, S., Geise, R., Gould, B., Keat, N., Huda, W., Judy, P., Kalender, W., McNitt-Gray, M., Morin, R., Payne, T., Stern, S., Rothenberg, L., Shrimpton, P., Timmer, J. & Wilson, C. (2008), The Measurement, Reporting, and Management of Radiation Dose in CT, Report AAPM No. 96, American Association of Physicists in Medicine (AAPM).

McCollough, C. H. & Schueler, B. A. (2000), 'Calculation of effective dose', *Med. Phys.* **27**(5), 828–837.

McNitt-Gray, M. F. (2002), 'AAPM/RSNA Physics Tutorial for Residents: Topics in CT. Radiation dose in CT', *Radiographics.* **22**(6), 1541–1553.

Metropolis, N. (1987), 'The Beginning ofthe Monte Carlo Method', *Los Alamos Science* (15), 125–130.

Miras, H., Jiménez, R., Miras, C. & Gomà, C. (2013), 'CloudMC: A cloud computing application for Monte Carlo simulation', *Phys. Med. Biol.* **58**(8), N125.

Moore, G. E. (1998), 'Cramming more components onto integrated circuits', *Proc. IEEE* **86**(1), 82–85.

Moore, S. K. (2010), Multicore CPUs: Processor proliferation, Technical report, IEEE Spectrum. `http://spectrum.ieee.org/semiconductors/processors/multicore-cpus-processor-proliferation` (Retrieved on July 09, 2014).

Na, Y. H., Zhang, B., Zhang, J., Caracappa, P. F. & Xu, X. G. (2010), 'Deformable adult human phantoms for radiation protection dosimetry: Anthropometric data representing size distributions of adult worker populations and software algorithms', *Phys. Med. Biol.* **55**(13), 3789–3811.

National Research Council (2005), *Health risks from exposure to low levels of ionizing radiation: BEIR VII-Phase 2*, Washington, D.C.

NCRP (2009), Ionizing radiation exposure of the population of the United States, Report NCRP 160, National Council on Radiation Protection & Measurements (NCRP).

NEMA (1996), Digital imaging and communications in medicine (DICOM), Technical report, National Electrical Manufacturers Association (NEMA).

Newegg. (2014*a*), Intel Xeon X5650 CPU price. `http://www.newegg.com/Product/Product.aspx?Item=N82E16819117231` (Retrieved on July 09, 2014).

Newegg. (2014*b*), Nvidia Tesla K40 GPU price. `http://www.newegg.com/Product/Product.aspx?Item=N82E16814132015` (Retrieved on July 09, 2014).

NIH. (2012), Decreasing Patient Radiation Dose from CT Imaging: Achieving Sub-mSv Studies (U01). `http://grants.nih.gov/grants/guide/pa-files/PAR-12-206.html` (Retrieved on July 09, 2014).

Nowotny, R. & Höfer, A. (1985), 'Ein Programm für die Berechnung von diagnostischen Röntgenspektren', *Fortschr Röntgenstr* **142**(6), 685–689.

Nvidia. (2011), Nvidia Next Generation CUDA Compute Architecture: Fermi. `http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf` (Retrieved on July 09, 2014).

Nvidia. (2012), CUDA CURAND guide. `http://docs.nvidia.com/cuda/curand/index.html` (Retrieved on July 09, 2014).

Nvidia. (2013*a*), CUDA C best practices guide. http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html (Retrieved on July 09, 2014).

Nvidia. (2013*b*), CUDA C programming guide. http://docs.nvidia.com/cuda/cuda-c-programming-guide/ (Retrieved on July 09, 2014).

Nvidia. (2013*c*), GPU occupancy calculator. http://developer.download.nvidia.com/compute/cuda/CUDA_Occupancy_calculator.xls (Retrieved on July 09, 2014).

Nvidia. (2013*d*), Kepler tuning guide. http://docs.nvidia.com/cuda/kepler-tuning-guide/index.html (Retrieved on July 09, 2014).

Nvidia. (2013*e*), Nvidia CUDA compiler driver nvcc. http://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html (Retrieved on July 09, 2014).

Nvidia. (2013*f*), NVIDIA System Management Interface program. http://developer.download.nvidia.com/compute/cuda/5_5/rel/nvml/nvidia-smi.5.319.43.pdf (Retrieved on July 09, 2014).

Nvidia. (2013*g*), Profiler user's guide. http://docs.nvidia.com/cuda/pdf/CUDA_Profiler_Users_Guide.pdf (Retrieved on July 09, 2014).

Nvidia. (2014), Nvidia Nsight Visual Studio Edition 4.0 user guide. https://developer.nvidia.com/nsight-visual-studio-edition-documentation-and-support (Retrieved on July 09, 2014).

Overberg, M., Moretti, B., Slovacek, R. & Block, R. (1999), 'Photoneutron target development for the RPI linear accelerator', *Nucl. Instrum. Meth. A* **438**(2), 253–264.

Panneton, F. & L'ecuyer, P. (2005), 'On the xorshift random number generators', *ACM TOMACS* **15**, 346–361.

Paraview. (2014), Paraview users' guide. http://www.paraview.org/Wiki/ParaView/Users_Guide/Table_Of_Contents (Retrieved on July 09, 2014).

Pearce, M. S., Salotti, J. A., Little, M. P., McHugh, K., Lee, C., Kim, K. P., Howe, N. L., Ronckers, C. M., Rajaraman, P., Craft, A. W., Parker, L. & González, A. B. d. (2012), 'Radiation exposure from CT scans in childhood and subsequent risk of leukaemia and brain tumours: A retrospective cohort study', *The Lancet* **380**(9840), 499–505.

Pelowitz, D. B. (2008), MCNPX user's manual, version 2.6.0, Report LA-CP-07-1473, Los Alamos National Laboratory (LANL).

Pelowitz, D. B. (2013*a*), MCNP6 release 1.0, verification and validation testing, Report LA-UR-13-xxxxx, Los Alamos National Laboratory (LANL).

Pelowitz, D. B. (2013*b*), MCNP6 user's manual, version 1.0, Report LA-CP-13-00634, Los Alamos National Laboratory (LANL).

Phillips, E. & Fatica, M. (2010), CUDA accelerated Linpack on clusters, *in* 'GPU Technology Conference (GTC) 2010', San Jose, CA.

Purches, J. (2013), Nvidia GPU technology. `https://intranet.birmingham.ac.uk/it/teams/infrastructure/fm/bear/documents/public/CUDA-2013-07-31/NVIDIA-Technology-Overview.pdf` (Retrieved on July 09, 2014).

Radcal. (2014), Radcal ion chamber energy dependence graphs. `http://www.radcal.com/pdf/Ion-Chamber-Energy-Dependence-Graphs.pdf` (Retrieved on July 09, 2014).

Ramey, C. & Fox, B. (2010), Bash reference manual. `http://www.gnu.org/software/bash/manual/bashref.html` (Retrieved on July 09, 2014).

Rennich, S. (2011), CUDA C/C++ streams and concurrency. `http://on-demand.gputechconf.com/gtc-express/2011/presentations/StreamsAndConcurrencyWebinar.pdf` (Retrieved on July 09, 2014).

Robert McNeel & Associates. (2014), Rhinoceros 5 user's guide for Windows. `http://www.rhino3d.com/download/rhino/5.0/UsersGuide` (Retrieved on July 09, 2014).

Rogers, D. W. (2006), 'Fifty years of Monte Carlo simulations for med. phys.', *Phys. Med. Biol.* **51**(13), 287–301.

Romano, P. K. & Forget, B. (2013), 'The OpenMC Monte Carlo particle transport code', *Ann. Nucl. Energy.* **51**, 274–281.

Schlattl, H., Zankl, M. & Petoussi-Henss, N. (2007), 'Organ dose conversion coefficients for voxel models of the reference male and female from idealized photon exposures', *Phys. Med. Biol.* **52**(8), 2123–2145.

Schneider, W., Bortfeld, T. & Schlegel, W. (2000), 'Correlation between CT numbers and tissue parameters needed for Monte Carlo simulations of clinical dose distributions', *Phys. Med. Biol.* **45**(2), 459–478.

Seibert. (2011), Nvidia developer zone, CUDA programming and performance: Newbie confusion: Thread, block, multiprocessor and processor. `https://devtalk.nvidia.com/default/topic/491744/newbie-confusion-thread-block-multiprocessor-and-processor/` (Retrieved on July 09, 2014).

Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J. & Cavin, R. (2008), 'Larrabee: A many-core x86 architecture for visual computing', *ACM TOG* **27**, 18.

Selcow, E. & McKinney, G. (2000), MCNP capabilities at the dawn of the 21st century: Neutron-gamma applications, *in* 'Advanced Monte Carlo for Radiation Physics, Particle Transport Simulation and Applications', Springer Berlin Heidelberg, Lisbon, Portugal, pp. 643–650.

Sempau, J., Wilderman, S. J. & Bielajew, A. F. (2000), 'DPM, a fast, accurate Monte Carlo code optimized for photon and electron radiotherapy treatment planning dose calculations', *Phys. Med. Biol.* **45**(8), 2263–2291.

Serov, I., John, T. & Hoogenboom, J. (1998), 'A new effective Monte Carlo midway coupling method in MCNP applied to a well logging problem', *Appl. Radiat. Isot.* **49**(12), 1737–1744.

Shammas, N. (1993), *Windows batch file programming*, Windcrest/McGraw-Hill, Blue Ridge Summit, PA.

SPEC. (2014), Standard Performance Evaluation Corporation (spec) power benchmark for intel xeon x5650 cpu. `http://www.spec.org/power_ssj2008/results/res2011q4/power_ssj2008-20111128-00414.txt` (Retrieved on July 09, 2014).

Stallman, R. M. (2003), Using the GNU compiler collection: For GCC version 4.4.7. `https://gcc.gnu.org/onlinedocs/gcc-4.4.7/gcc.pdf` (Retrieved on July, 09, 2014).

Stallman, R. M., McGrath, R. & Smith, P. D. (2013), GNU Make. `http://www.gnu.org/software/make/manual/make.html` (Retrieved on July 09, 2014).

Stern, S. H. (2007), Nationwide evaluation of X-ray trends (NEXT) tabulation and graphical summary of 2000 survey of Computed Tomography, Report CRCPD E-07-2, Conference of Radiation Control Program Directors, Inc. (CRCPD).

Stern, S. H., Spelic, D. C. & Kaczmarek, R. V. (2000), NEXT 2000 protocol for survey of Computed Tomography (CT), Technical report, Conference of Radiation Control Program Directors, Inc. (CRCPD).

Su, L., Du, X, L. T. & Xu, X. G. (2013), GPU-Accelerated Monte Carlo Electron Transport Methods: Development and Applicationfor Radiation Dose Calculations Using Six GPU Cards, *in* 'Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo (SNA & MC 2013)', American Nuclear Society (ANS), Paris, France.

Su, L., Yang, Y., Bednarz, B., Sterpin, E., Du, X., Liu, T., Ji, W. & Xu, X. G. (2014), 'ARCHER$_{\mathrm{RT}}$, A photon-electron coupled Monte Carlo dose computing engine for GPU: Software development of and application to helical tomotherapy', *Med. Phys.* **41**(7), 071709.

Top500. (2013), Top 500 supercomputer sites. `http://www.top500.org/lists/` (Retrieved on July 09, 2014).

TYAN. (2010), TYAN FT77-B7015 service engineer's manual. http://www.tyan.com/manuals/FT77-B7015_Manual.pdf (Retrieved on July 09, 2014).

US Energy Information Administration. (2014), Average retail price of electricity to ultimate customers by end-use sector. http://www.eia.gov/electricity/monthly/epm_table_grapher.cfm?t=epmt_5_6_a (Retrieved on July 09, 2014).

van Heesch, D. (2008), Doxygen: Source code documentation generator tool. http://www.stack.nl/~dimitri/doxygen/ (Retrieved on July 09, 2014).

van Rossum, G. (2014), The Python language reference. https://docs.python.org/2/reference/ (Retrieved on July 09, 2014).

Virtual Phantoms Inc. (2013), Virtual Phantoms web-based software for CT organ dose calculations. http://www.virtualphantoms.com/ (Retrieved on July 09, 2014).

Wang, B., Xu, X. G. & Kim, C. H. (2004), 'A Monte Carlo CT Model of the Rando Phantom', *Trans. Am. Nucl. Soc.* **90**, 473–474.

Wang, H., Ma, Y., Pratx, G. & Xing, L. (2011), 'Toward real-time Monte Carlo simulation using a commercial cloud computing infrastructure', *Phys. Med. Biol.* **56**(17), N175.

Wavefront Technologies. (2013), Wavefront obj specifications. http://www.martinreddy.net/gfx/3d/OBJ.spec (Retrieved on July 09, 2014).

Wechser, O. (2014), Developing the HPC compute cores of tomorrow. http://events.prace-ri.eu/getFile.py/access?contribId=8&sessionId=0&resId=0&materialId=slides&confId=176 (Retrieved on July 09, 2014).

Woo, M., Neider, J., Davis, T. & Shreiner, D. (1999), *OpenGL programming guide: The official guide to learning OpenGL, version 1.2*, Addison-Wesley Longman Publishing Co., Inc.

Woodcock, E. R., Murphy, T., Hemmings, P. J. & Longworth, T. C. (1965), 'Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry', *Proc. Conf. Applications of Computing Methods to Reactor Problems* **557**.

Wu, Y. (2006), 'Conceptual design activities of FDS series fusion power plants in China', *Fusion Engineering and Design* **81**(23), 2713–2718.

X-5 Monte Carlo Team (2003*a*), MCNP-a general Monte Carlo N-Particle Transport code, Version 5. Volume I: Overview and theory, Report LA-UR-03-1987, Los Alamos National Laboratory (LANL).

X-5 Monte Carlo Team (2003*b*), MCNP-a general Monte Carlo N-Particle Transport code, Version 5. Volume II: User's guide, Report LA-CP-03-0245, Los Alamos National Laboratory (LANL).

Xu, X. G., Bednarz, B. & Paganetti, H. (2008), 'A review of dosimetry studies on external-beam radiation treatment with respect to second cancer induction', *Phys. Med. Biol.* **53**(13), R193–R241.

Xu, X. G., Chao, T. C. & Bozkurt, A. (2000), 'VIP-Man: an image-based whole-body adult male model constructed from color photographs of the Visible Human Project for multi-particle Monte Carlo calculations', *Health Phys.* **78**(5), 476–486.

Xu, X. G. & Eckerma, K. F. (2010), *Handbook of anatomical models for radiation dosimetry*, CRC Press, Boca Raton, FL.

Xu, X. G., Liu, T., Su, L., Du, X., Riblett, M., Ji, W. & Brown, F. B. (2013), 'An update of ARCHER, a Monte Carlo radiation transport software testbed for emerging hardware such as GPUs', *Trans. Am. Nucl. Soc.* **108**, 433–434.

Xu, X. G., Taranenko, V., Zhang, J. & Shi, C. (2007), 'A boundary-representation method for designing whole-body radiation dosimetry models: Pregnant females at the ends of three gestational periods–RPI-P3, -P6 and -P9', *Phys. Med. Biol.* **52**(23), 7023–7044.

Zhang, D., Padole, A., Li, X., Singh, S., Khawaja, R. D. A., Lira, D., Liu, T., Shi, J. Q., Otrakji, A., Kalra, M. K., Xu, X. G. & Liu, B. (2014), 'In vitro dose measurements in a human cadaver with abdomen/pelvis CT scans', *Med. Phys.* . (accepted).

Zhang, J., Na, Y. H., Caracappa, P. F. & G, X. X. (2009), 'RPI-AM and RPI-AF, 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**(19), 5885–5908.

APPENDIX A
VERIFICATION OF ARCHER WITH MCNP

A.1 73 kg RPI adult male phantom

Table A.1: Verification of ARCHER with MCNP using 73 kg RPI adult male phantom.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
colon wall	7.20e-08	6.16e-04	7.21e-08	8.91e-04	0.20%
lungs (left & right)	7.94e-08	3.39e-04	7.97e-08	4.75e-04	0.28%
stomach wall	7.23e-08	8.27e-04	7.25e-08	1.29e-03	0.19%
breast (left & right)	6.93e-08	1.59e-03	6.90e-08	2.84e-03	-0.45%
gonads for male	1.16e-07	1.50e-03	1.16e-07	2.21e-03	0.13%
urinary bladder wall	6.06e-08	1.23e-03	6.07e-08	2.29e-03	0.13%
esophagus	8.41e-08	1.23e-03	8.42e-08	2.19e-03	0.11%
liver	8.35e-08	4.68e-04	8.35e-08	5.09e-04	0.08%
thyroid	1.42e-07	1.59e-03	1.40e-07	2.56e-03	-0.80%
brain	8.96e-08	5.28e-04	8.97e-08	5.67e-04	0.16%
salivary glands (left & right)	1.03e-07	8.62e-04	1.03e-07	1.40e-03	0.21%
skin	9.17e-08	9.18e-05	9.19e-08	2.33e-04	0.26%
adrenals	7.93e-08	2.20e-03	7.95e-08	4.00e-03	0.26%
extrathoracic region	8.78e-08	2.01e-03	8.81e-08	3.07e-03	0.37%
gall bladder wall	6.21e-08	1.95e-03	6.22e-08	4.65e-03	0.07%
heart wall	7.48e-08	6.81e-04	7.50e-08	9.33e-04	0.24%
kidneys (left & right)	8.00e-08	8.32e-04	8.01e-08	1.04e-03	0.12%
lymphatic nodes	6.72e-08	6.80e-04	6.72e-08	1.36e-03	0.03%
muscle	8.64e-08	1.13e-04	8.65e-08	1.27e-04	0.16%

Table A.1: Continued.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
oral mucosa	8.77e-08	1.66e-03	8.80e-08	2.62e-03	0.39%
pancreas	6.61e-08	1.24e-03	6.62e-08	1.58e-03	0.04%
prostate	7.14e-08	2.55e-03	7.08e-08	3.98e-03	-0.87%
small intestine	7.05e-08	6.33e-04	7.05e-08	7.08e-04	-0.08%
spleen	8.23e-08	1.12e-03	8.25e-08	1.44e-03	0.17%
thymus	7.94e-08	1.76e-03	7.96e-08	2.89e-03	0.28%
spongiosa combined	1.09e-07	1.85e-04	1.09e-07	2.34e-04	-0.17%
medullary cavity combined	6.60e-08	6.15e-04	6.63e-08	1.08e-03	0.36%
cortical bone combined	2.90e-07	1.79e-04	2.89e-07	2.06e-04	-0.48%
all bone combined	1.83e-07	1.59e-04	1.82e-07	1.70e-04	-0.37%
adipose tissue	6.87e-08	1.31e-04	6.88e-08	1.70e-04	0.14%
humeri, upper half, spongiosa	4.92e-08	1.17e-03	4.91e-08	1.56e-03	-0.27%
clavicles, spongiosa	5.63e-08	1.14e-03	5.61e-08	2.04e-03	-0.32%
cranium, spongiosa	7.51e-08	4.58e-04	7.49e-08	6.64e-04	-0.37%
femora, upper half, spongiosa	5.85e-08	6.20e-04	5.82e-08	8.06e-04	-0.51%
mandible, spongiosa	8.01e-08	9.92e-04	7.96e-08	1.51e-03	-0.59%
pelvis, spongiosa	4.91e-08	5.79e-04	4.88e-08	7.38e-04	-0.55%
ribs, spongiosa	6.40e-08	3.72e-04	6.37e-08	6.23e-04	-0.55%
scapulae, spongiosa	5.07e-08	9.61e-04	5.05e-08	1.29e-03	-0.52%
cervical spine, spongiosa	8.51e-08	9.54e-04	8.50e-08	1.56e-03	-0.14%
thoracic spine, spongiosa	6.00e-08	7.09e-04	5.98e-08	9.59e-04	-0.38%
lumbar spine, spongiosa	4.86e-08	8.78e-04	4.85e-08	1.12e-03	-0.37%
sacrum, spongiosa	5.48e-08	1.03e-03	5.48e-08	1.42e-03	-0.12%
sternum, spongiosa	8.39e-08	1.06e-03	8.37e-08	1.80e-03	-0.15%

Table A.1: Continued.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
all, spongiosa	5.93e-08	2.19e-04	5.91e-08	2.89e-04	-0.43%

A.2 142 kg RPI adult male phantom

Table A.2: Verification of ARCHER with MCNP using 142 kg RPI adult male phantom.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
colon wall	3.21e-08	9.53e-04	3.20e-08	1.35e-03	-0.30%
lungs (left & right)	5.86e-08	4.03e-04	5.87e-08	5.57e-04	0.09%
stomach wall	3.54e-08	1.22e-03	3.54e-08	1.86e-03	-0.08%
breast (left & right)	5.41e-08	1.77e-03	5.46e-08	3.22e-03	0.95%
gonads for male	9.40e-08	1.68e-03	9.44e-08	2.46e-03	0.44%
urinary bladder wall	2.78e-08	1.85e-03	2.77e-08	3.41e-03	-0.45%
esophagus	7.38e-08	1.32e-03	7.38e-08	2.35e-03	-0.02%
liver	4.49e-08	6.52e-04	4.50e-08	7.02e-04	0.17%
thyroid	1.31e-07	1.68e-03	1.30e-07	2.67e-03	-0.92%
brain	8.86e-08	5.11e-04	8.88e-08	5.71e-04	0.23%
salivary glands (left & right)	1.00e-07	8.91e-04	1.01e-07	1.43e-03	0.48%
skin	8.05e-08	8.59e-05	8.08e-08	2.09e-04	0.35%
adrenals	5.79e-08	2.74e-03	5.75e-08	4.74e-03	-0.70%
extrathoracic region	8.34e-08	2.05e-03	8.38e-08	3.16e-03	0.48%
gall bladder wall	2.93e-08	2.94e-03	2.96e-08	6.78e-03	1.02%
heart wall	5.20e-08	8.17e-04	5.20e-08	1.12e-03	-0.09%
kidneys (left & right)	4.48e-08	1.19e-03	4.47e-08	1.43e-03	-0.17%
lymphatic nodes	4.90e-08	7.76e-04	4.90e-08	1.59e-03	0.02%

Table A.2: Continued.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
muscle	6.29e-08	1.35e-04	6.30e-08	1.52e-04	0.17%
oral mucosa	8.94e-08	1.67e-03	8.96e-08	2.59e-03	0.21%
pancreas	2.72e-08	2.01e-03	2.70e-08	2.52e-03	-0.49%
prostate	3.61e-08	3.65e-03	3.55e-08	5.63e-03	-1.67%
small intestine	2.66e-08	1.06e-03	2.65e-08	1.16e-03	-0.27%
spleen	5.45e-08	1.44e-03	5.45e-08	1.80e-03	-0.10%
thymus	6.12e-08	2.02e-03	6.12e-08	3.31e-03	0.02%
spongiosa combined	8.52e-08	2.08e-04	8.52e-08	2.64e-04	0.08%
medullary cavity combined	5.07e-08	7.12e-04	5.08e-08	1.23e-03	0.06%
cortical bone combined	2.34e-07	2.01e-04	2.35e-07	2.29e-04	0.15%
all bone combined	1.46e-07	1.79e-04	1.46e-07	1.90e-04	0.12%
adipose tissue	5.42e-08	9.79e-05	5.44e-08	1.07e-04	0.34%
humeri, upper half, spongiosa	3.87e-08	1.12e-03	3.86e-08	1.51e-03	-0.40%
clavicles, spongiosa	4.83e-08	1.17e-03	4.82e-08	2.18e-03	-0.13%
cranium, spongiosa	7.42e-08	4.54e-04	7.42e-08	6.69e-04	0.01%
femora, upper half, spongiosa	3.21e-08	8.43e-04	3.21e-08	1.07e-03	-0.15%
mandible, spongiosa	7.81e-08	9.84e-04	7.81e-08	1.51e-03	-0.06%
pelvis, spongiosa	2.46e-08	8.15e-04	2.45e-08	1.03e-03	-0.55%
ribs, spongiosa	5.00e-08	4.22e-04	5.00e-08	7.00e-04	-0.03%
scapulae, spongiosa	4.26e-08	1.04e-03	4.25e-08	1.40e-03	-0.17%
cervical spine, spongiosa	7.71e-08	9.95e-04	7.72e-08	1.63e-03	0.09%
thoracic spine, spongiosa	4.75e-08	7.92e-04	4.74e-08	1.07e-03	-0.26%
lumbar spine, spongiosa	2.26e-08	1.28e-03	2.24e-08	1.62e-03	-0.64%
sacrum, spongiosa	2.90e-08	1.42e-03	2.91e-08	1.95e-03	0.17%

Table A.2: Continued.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
sternum, spongiosa	6.20e-08	1.27e-03	6.18e-08	2.10e-03	-0.27%
all, spongiosa	4.24e-08	2.50e-04	4.23e-08	3.33e-04	-0.16%

A.3 122 kg RPI adult female phantom

Table A.3: Verification of ARCHER with MCNP using 122 kg RPI adult female phantom.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
colon wall	3.60e-08	9.46e-04	3.59e-08	1.34e-03	-0.19%
lungs (left & right)	6.75e-08	4.31e-04	6.75e-08	6.03e-04	0.00%
stomach wall	4.00e-08	1.26e-03	4.00e-08	1.89e-03	-0.16%
breast (left & right)	5.50e-08	7.43e-04	5.52e-08	9.46e-04	0.42%
ovaries for female	2.96e-08	4.20e-03	2.98e-08	7.39e-03	0.72%
urinary bladder wall	3.53e-08	1.80e-03	3.52e-08	3.46e-03	-0.10%
esophagus	8.18e-08	1.37e-03	8.20e-08	2.46e-03	0.16%
liver	5.40e-08	6.80e-04	5.41e-08	7.39e-04	0.18%
thyroid	1.37e-07	1.78e-03	1.36e-07	2.92e-03	-1.02%
brain	9.82e-08	5.32e-04	9.85e-08	5.90e-04	0.23%
salivary glands (left & right)	1.06e-07	1.03e-03	1.07e-07	1.61e-03	0.31%
skin	8.81e-08	8.99e-05	8.85e-08	2.12e-04	0.37%
adrenals	6.00e-08	2.90e-03	6.03e-08	4.92e-03	0.40%
extrathoracic region	1.20e-07	1.97e-03	1.20e-07	3.06e-03	0.12%
gall bladder wall	3.34e-08	3.15e-03	3.36e-08	7.38e-03	0.49%
heart wall	5.89e-08	8.90e-04	5.87e-08	1.25e-03	-0.26%
kidneys (left & right)	4.65e-08	1.27e-03	4.65e-08	1.53e-03	-0.00%

Table A.3: Continued.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
lymphatic nodes	5.81e-08	9.91e-04	5.79e-08	1.97e-03	-0.27%
muscle	7.12e-08	1.52e-04	7.13e-08	1.76e-04	0.13%
oral mucosa	1.16e-07	2.47e-03	1.15e-07	4.21e-03	-0.55%
pancreas	2.86e-08	2.20e-03	2.86e-08	2.75e-03	-0.24%
uterus	2.43e-08	2.60e-03	2.42e-08	3.47e-03	-0.38%
small intestine	2.93e-08	1.11e-03	2.92e-08	1.22e-03	-0.41%
spleen	6.71e-08	1.41e-03	6.73e-08	1.78e-03	0.31%
thymus	6.78e-08	2.15e-03	6.77e-08	3.63e-03	-0.18%
spongiosa combined	9.79e-08	2.32e-04	9.76e-08	3.06e-04	-0.23%
medullary cavity combined	5.92e-08	6.63e-04	5.91e-08	1.18e-03	-0.16%
cortical bone combined	2.84e-07	2.11e-04	2.86e-07	2.49e-04	0.85%
all bone combined	1.79e-07	1.94e-04	1.79e-07	2.11e-04	-0.43%
adipose tissue	5.94e-08	9.85e-05	5.96e-08	1.05e-04	0.24%
humeri, upper half, spongiosa	4.27e-08	1.39e-03	4.24e-08	1.86e-03	-0.83%
clavicles, spongiosa	5.53e-08	1.22e-03	5.47e-08	2.41e-03	-1.03%
cranium, spongiosa	8.35e-08	4.75e-04	8.32e-08	7.04e-04	-0.36%
femora, upper half, spongiosa	3.51e-08	1.07e-03	3.47e-08	1.44e-03	-0.96%
mandible, spongiosa	9.49e-08	1.14e-03	9.40e-08	1.95e-03	-0.98%
pelvis, spongiosa	3.02e-08	8.71e-04	2.99e-08	1.15e-03	-0.86%
ribs, spongiosa	5.97e-08	4.63e-04	5.94e-08	8.73e-04	-0.60%
scapulae, spongiosa	4.74e-08	1.24e-03	4.70e-08	1.79e-03	-0.74%
cervical spine, spongiosa	9.03e-08	9.97e-04	9.03e-08	1.64e-03	-0.05%
thoracic spine, spongiosa	5.14e-08	8.41e-04	5.11e-08	1.19e-03	-0.55%
lumbar spine, spongiosa	2.67e-08	1.30e-03	2.65e-08	1.68e-03	-0.94%

Table A.3: Continued.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
sacrum, spongiosa	3.87e-08	1.52e-03	3.87e-08	2.06e-03	0.08%
sternum, spongiosa	6.93e-08	1.34e-03	6.89e-08	2.27e-03	-0.61%
all, spongiosa	5.01e-08	2.74e-04	4.98e-08	3.81e-04	-0.58%

A.4 RPI 9-month pregnant female phantom

Table A.4: Verification of ARCHER with MCNP using RPI 9-month pregnant female phantom.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
brain	7.50e-08	6.01e-04	7.51e-08	6.69e-04	0.19%
eyeballs	1.01e-07	1.91e-03	1.01e-07	3.43e-03	-0.24%
eye lens	1.13e-07	5.53e-03	1.11e-07	1.74e-02	-1.92%
thyroid	1.35e-07	1.67e-03	1.34e-07	2.90e-03	-0.46%
trachea	8.43e-08	1.97e-03	8.35e-08	4.72e-03	-0.88%
thymus	8.48e-08	1.91e-03	8.54e-08	3.26e-03	0.71%
lungs	7.73e-08	3.81e-04	7.74e-08	5.56e-04	0.14%
heart wall	7.77e-08	8.06e-04	7.78e-08	1.11e-03	0.12%
esophagus wall	7.29e-08	1.40e-03	7.30e-08	2.60e-03	0.18%
breasts	8.84e-08	4.87e-04	8.85e-08	6.02e-04	0.19%
stomach wall	6.33e-08	9.95e-04	6.35e-08	1.51e-03	0.27%
liver	7.25e-08	5.99e-04	7.25e-08	6.52e-04	-0.06%
gallbladder wall	5.88e-08	2.17e-03	5.78e-08	5.58e-03	-1.68%
pancreas	5.33e-08	1.54e-03	5.30e-08	2.00e-03	-0.49%
spleen	7.75e-08	1.30e-03	7.77e-08	1.66e-03	0.19%
kidneys	6.99e-08	9.29e-04	6.99e-08	1.20e-03	0.10%
adrenals	6.39e-08	2.33e-03	6.41e-08	4.42e-03	0.36%

Table A.4: Continued.

organ	MCNP		ARCHER		difference
	dose	RSD	dose	RSD	[%]
small intestine wall and contents	5.21e-08	6.07e-04	5.20e-08	7.72e-04	-0.11%
large intestine wall	5.58e-08	6.46e-04	5.58e-08	9.98e-04	-0.04%
large intestine contents	5.74e-08	7.57e-04	5.72e-08	1.10e-03	-0.25%
ovaries	3.97e-08	3.55e-03	3.98e-08	6.38e-03	0.05%
bladder wall	6.13e-08	1.33e-03	6.11e-08	2.63e-03	-0.40%
uterine wall	6.81e-08	3.51e-04	6.82e-08	5.29e-04	0.13%
uterine contents	5.75e-08	3.85e-04	5.74e-08	4.19e-04	-0.05%
placenta	7.67e-08	6.23e-04	7.68e-08	7.67e-04	0.17%
fetal soft tissue	5.91e-08	4.53e-04	5.91e-08	4.99e-04	-0.08%
fetal skeleton	1.71e-07	6.82e-04	1.70e-07	8.40e-04	-0.75%
fetal brain	3.98e-08	1.30e-03	3.96e-08	1.53e-03	-0.56%
skeleton	2.55e-07	1.64e-04	2.54e-07	1.72e-04	-0.58%
skin	8.80e-08	9.50e-05	8.82e-08	2.68e-04	0.23%
remainder	7.70e-08	9.89e-05	7.71e-08	1.09e-04	0.15%
fetus total	5.68e-08	4.44e-04	5.67e-08	4.85e-04	-0.12%

APPENDIX B
LIST OF JOURNAL AND CONFERENCE PAPERS

My doctoral research at RPI has covered a wide spectrum of topics. This thesis is related to the following peer-reviewed journal and conference papers.

B.1 Journals

Liu, T., Du, X., Su, L., Ji, W., Carothers, C. D., Shephard, M. S., Liu, B., Kalra, M. K., Brown, F. B., Fitzgerald, P. F. & Xu, X. G. (2014), 'ARCHER-CT, an extremely fast Monte Carlo code for patient-specific ct dose calculations using Nvidia GPU and Intel coprocessor technologies: part I — software development and testing', *Phys. Med. Biol.* (submitted).

Zhang, D., Padole, A., Li, X., Singh, S., Khawaja, R. D. A., Lira, D., **Liu, T.**, Shi, J. Q., Otrakji, A., Kalra, M. K., Xu, X. G. & Liu, B. (2014), 'In vitro dose measurements in a human cadaver with abdomen/pelvis CT scans', *Med. Phys.* (accepted).

Su, L., Yang, Y., Bednarz, B., Sterpin, E., Du, X., **Liu, T.**, Ji, W. & Xu, X. G. (2014), 'ARCHER-RT, A photon-electron coupled Monte Carlo dose computing engine for GPU: software development of and application to helical tomotherapy', *Med. Phys.* **41**(7), 071709.

B.2 Conference Abstracts and Papers

Liu, T., Su, L., Du, X., Lin, H., Zieb, K., Ji, W., Caracappa, P. & Xu, X. G. (2014), Parallel Monte Carlo methods for heterogeneous hardware computer systems using GPUs and coprocessors: recent development of ARCHER code, *in* '18th Topical Meeting of the Radiation Protection and Shielding Division (RPSD) of the American Nuclear Society', Knoxville, TN.

Liu, T., Su, L., Du, X., Caracappa, P. F. & Xu, X. G. (2014), Comparison of accuracy and speed of ARCHER with MCNP for organ dose calculations from

external photon beams under standard irradiation geometries, *in* '59th Annual Meeting of the Health Physics Society', Baltimore, MD.

Liu, T., Du, X., Su, L., Ji, W. & Xu, X. G. (2014), Development of ARCHER-CT, a fast Monte Carlo code for patient-specific CT dose calculations using Nvidia GPU and Intel coprocessor technologies, *in* 'GPU Technology Conference 2014', San Jose, CA.

Liu, T., Du, X., Su, L., Gao, Y., Ji, W., Zhang, D., Shi, J. Q., Liu, B., Kalra, M. K. & Xu, X. G. (2014), Monte Carlo CT dose calculation: a comparison between experiment and simulation using ARCHER-CT, *in* 'AAPM 56th Annual Meeting & Exhibition', Austin, TX.

Liu, T., Du, X., Su, L., Gao, Y., Ji, W., Zhang, D., Shi, J. Q., Liu, B., Kalra, M. K. & Xu, X. G. (2014), Testing of ARCHER-CT, a fast Monte Carlo Code for CT dose calculation: experiment versus simulation, *in* 'American Nuclear Society (ANS) 2014 Annual Meeting', Reno, NV.

Liu, T., Du, X., Ji, W., Xu, X. G. & Brown, F. B. (2013), A comparative study of history-based versus vectorized Monte Carlo methods in the GPU/CUDA environment for a simple neutron eigenvalue problem, *in* 'Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo (SNA & MC 2013)', Paris, France.

Su L, Du X, **Liu, T.** & Xu, X. G. (2013), GPU-accelerated Monte Carlo electron transport methods: development and application for radiation dose calculations using six GPU cards, *in* 'Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo (SNA & MC 2013)', Paris, France.

Xu, X. G., **Liu, T.**, Su, L., Du, X., Riblett, M. J., Ji, W., Gu, D., Carothers, C. D., Shephard, M. S., Brown, F. B., Kalra, M. K. & Liu, B. (2013), ARCHER, a new Monte Carlo software tool for emerging heterogeneous computing environments, *in* 'Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo (SNA & MC 2013)', Paris, France.

Liu, T., Ji, W. & Xu, X. G. (2013), Development of GPU-based Monte Carlo code for fast CT imaging dose calculation on CUDA Fermi architecture, *in* 'International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 13)', Sun Valley, ID.

Liu, T., Xu, X. G. & Carothers, C. D. (2013), Comparison of two accelerators for Monte Carlo radiation transport calculations, NVIDIA Tesla M2090 GPU and Intel Xeon Phi 5110p coprocessor: a case study for X-ray CT imaging dose calculation, *in* 'Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo (SNA & MC 2013)', Paris, France.

Du, X., **Liu, T.**, Su, L., Riblett, M. & Xu, X. G. (2013), A hardware accelerator based fast Monte Carlo code for radiation dosimetry: software design and preliminary results, *in* 'AAPM 55th Annual Meeting & Exhibition', Indianapolis, IN.

Liu, T., Du, X. & Xu, X. G. (2013), Affordable supercomputer-based Monte Carlo CT dose calculations: a hardware comparison between Nvidia M2090 GPU and Intel Xeon Phi 5110p coprocessor, *in* 'AAPM 55th Annual Meeting & Exhibition', Indianapolis, IN.

Riblett, M. J., **Liu, T.**, Ji, W. & Xu, X. G. (2013), Use of hardware accelerators for Monte Carlo-based neutron radiation transport: a preliminary study, *in* '58th Annual Meeting of the Health Physics Society', Madison, WI.

Su, L., Du, X., **Liu, T.** & Xu, X. G. (2013), Fast Monte Carlo electron-photon transport code using hardware accelerators: preliminary results for brachytherapy and radionuclide therapy cases, *in* 'AAPM 55th Annual Meeting & Exhibition', Indianapolis, IN.

Su, L., Du, X., **Liu, T.** & Xu, X. G. (2013), A fast Monte Carlo electron transport code for dose calculations using the GPU accelerator, *in* '58th Annual Meeting of the Health Physics Society', Madison, WI.

Xu, X. G., **Liu, T.**, Su, L., Du, X., Riblett, M., Ji, W. & Brown, F. B. (2013), 'An update of ARCHER, a Monte Carlo radiation transport software testbed for emerging hardware such as GPUs', *Trans. Am. Nucl. Soc.* **108**, 433-434.

Liu, T., Ding, A., Ji, W., Xu, X. G., Carothers, C. D. & Brown, F. B. (2012), A Monte Carlo neutron transport code for eigenvalue calculations on a dual-GPU system and CUDA environment, *in* 'International Topical Meeting on Advances in Reactor Physics (PHYSOR 2012)', Knoxville, TN.

Liu, T., Ding, A. & Xu, X. G. (2012), 'GPU-based Monte Carlo methods for accelerating radiographic and CT imaging dose calculations: feasibility and scalability', *Med. Phys.* **39**, 3876.

Liu, T., Ding, A. & Xu, X. G. (2012), 'Accelerated Monte Carlo methods for photon dosimetry using a dual-GPU system and CUDA', *Med. Phys.* **39**, 3818.

Vazquez, J., Ding, A., **Liu, T.**, Su, L., Gao, Y., Mille, M., Caracappa, P. F. & Xu, X. G. (2012), Current research pursuits the Rensselaer Radiation Measurement and Dosimetry Group, *in* 'The American Nuclear Society 2012 Student Conference', Las Vegas, NV.

Xu, X. G., Su, L., **Liu, T.** & Ding, A. (2012), GPU-based Monte Carlo method for medical physics applications: preliminary results for X-ray and proton applications, *in* 'World Congress on Medical Physics and Biomedical Engineering (WC 2012)', Beijing, China.

Su, L., **Liu, T.**, Ding, A. & Xu, X. G. (2012), GPU/CUDA-based Monte Carlo methods for radiation protection dose calculations involving X-ray and proton sources, *in* '57th Annual Meeting of the Health Physics Society', Sacramento, CA.

Su, L., **Liu, T.**, Ding, A. & Xu, X. G. (2012), 'A GPU/CUDA based Monte Carlo code for proton transport: preliminary results of proton depth dose in water', *Med. Phys.* **39**, 3945.

Liu, T., Su, L., Ding, A., Ji, W., Carothers, C. D. & Xu, X. G. (2012), 'GPU/CUDA-ready parallel Monte Carlo codes for reactor analysis and other applications', *Trans. Am. Nucl. Soc.* **106**, 378-379.

Ding, A., **Liu, T.**, Liang, C., Ji, W., Shepard, M. S., Xu, X. G. & Brown, F. B. (2011), Evaluation of speedup of Monte Carlo calculations of simple reactor physics problems coded for the GPU/CUDA environment, *in* 'International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 11)', Rio de Janeiro, Brazil.