Towards real-time photon Monte Carlo dose calculation in the cloud

Towards real-time Photon Monte Carlo Dose Calculation in the Cloud

Peter Ziegenhein, Igor N. Kozin, Cornelis Ph. Kamerling and Uwe Oelfke

Joint Department of Physics at The Institute of Cancer Research and The Royal Marsden NHS Foundation Trust, London, UK SM2 5NG

E-mail: Peter.Ziegenhein@icr.ac.uk

Abstract. Near real-time application of Monte Carlo (MC) dose calculation in clinic and research is hindered by long computational runtimes of established software. Currently, fast MC software solutions are available utilising accelerators such as graphical processing units (GPUs) or clusters based on central processing units. Both platforms are expensive in terms of purchase costs and maintenance and, in case of the GPU, provide only limited scalability. In this work we propose a cloud-based MC solution, which offers high scalability of accurate photon dose calculations. The MC simulations run on a private virtual supercomputer that is formed in the cloud. Computational resources can be provisioned dynamically at low costs without upfront investment in expensive hardware. A client-server software solution has been developed which controls the simulations and transports data to and from the cloud efficiently and securely. The client application integrates seamlessly into a treatment planning system. It runs the MC simulation workflow automatically and securely exchanges simulation data with the server side application that controls the virtual supercomputer. The Advanced Encryption Standard was used to add an addition security layer which encrypts and decrypts patient data on-the-fly at the processor register level. We could show that our cloud-based MC framework enables near real-time dose computation. It delivers excellent linear scaling for high-resolution datasets with absolute runtimes of 1.1 to 10.9 seconds for simulating a clinical prostate and liver case up to 1% statistical uncertainty. The computation times include the data transportation to and from the cloud as well as process scheduling and synchronisation overhead. Cloud-based MC simulations offer a fast, affordable and easily accessible alternative for near real-time accurate dose calculations to currently used GPU or cluster solutions.

Submitted to: *Phys. Med. Biol.*

1. Introduction

Monte Carlo (MC) simulations are considered to be one of the most accurate dose calculation techniques for treatment planning in radiation therapy (RT). For some treatment modalities, e.g. in the presence of a strong magnetic field in an MR-Linac (Lagendijk et al.; 2008) it is currently the only reliable method to precisely estimate the influence of the magnetic field on the tracks of secondary electrons. For standard IMRT/VMAT treatment planning, commercial treatment planning systems (TPS) can perform MC simulations in the order of minutes which is suitable for most clinical indications. However, current simulation times are too high to use MC in adaptive radiation therapy (ART) scenarios in which dose calculation needs to be done in a certain time-frame.

Over the last years, various groups have developed fast MC dose calculation solutions on different computational platforms. Especially the utilisation of graphics processing units (GPUs) has been very popular in an attempt to reduce calculation times significantly. A number of GPU-based implementations have been introduced for photons, electrons and protons, for example (Jia et al.; 2011; Hissoiny et al.; 2011; Jia et al.; 2012; Jahnke et al.; 2012; Townson et al.; 2013). These groups claim speed-up factors of up to several 100× compared to central processing unit (CPU) implementations. In light of these overwhelmingly positive results, it comes as no surprise that very few MC implementations for CPU-based architectures have been developed in recent years (Su et al.; 2014; Tyagi et al.; 2004; Ziegenhein et al.; 2015).

A closer look into the GPU-based MC publications reveals that high speed-up factors are often reported in comparison to un-optimised CPU-based implementations. Other studies show that the performance advantage of GPUs is much smaller than anticipated (Lee et al.; 2010; Jia et al.; 2015) when compared to optimized CPU-based implementations. GPUs have their own physical memory which leads to two potential limitations: First, the size of that memory even on high-end GPUs is typically limited to a few gigabytes (GB) while powerful CPU-based systems can be populated with hundreds of GB of main memory. Second, due to the physical separation of main memory it is necessary to transport data to and from the GPU which may form a bottleneck for applications which run a short amount of time. Furthermore, special programming knowledge is required to use the performance potential of a GPU as well as the hardware itself which is constantly evolving. Tyagi et al. (2004) reported an almost linear performance scaling of the MC simulation on CPU-based clusters. However, this solution is hardly feasible for research and clinical applications due to the high cumulative cost of the cluster infrastructure, its maintenance and operational expenses. A recently published point/counterpoint discussion (Jia et al.; 2015) further investigates the question of which computational platform (GPUs or CPUs) is best suited to realise near real-time MC simulations for modern treatment planning applications. The authors came to the conclusion that there is no clear winner at the moment.

In this work, we discuss another potential solution for near real-time MC simulations:

cloud computing. Using computational resources in the cloud for MC has been previously investigated by (Pratx and Xing; 2011) and (Miras et al.; 2013). These authors could demonstrate an almost linear performance scaling if the simulation runtimes are high enough (in the order of minutes). However much shorter or even real-time overall dose calculation times could not be demonstrated. In the context of this work scalability will be denoted as *strong scaling*, which is conventionally defined as the relation between the runtime and the number of computational resources employed on a fixed size problem. Therefore, a strong scaling MC simulation framework should be able to deliver (almost) arbitrarily short runtimes for a specific dose calculation problem by using more and more computational resources (processors or computer nodes) in parallel. The performance scaling down to short runtimes in the order of a few seconds is limited by the simulation overhead which does not scale with the number of resources. In a cloud computing environment a substantially larger overhead may occur due to the need of transporting data up to the cloud, scheduling the calculation, obtaining the results and sending the dose distribution back to the client computer. The potentially large overhead is one of the main reasons why there are currently no cloud-based MC solutions delivering competitive overall calculation times.

In this paper, we present a highly integrated dose calculation framework for cloud and cluster computing which reduces the overhead significantly and provides excellent scaling down to an overall simulation time of only a few seconds. We demonstrate that scaling cloud resources can be used efficiently to deliver near real-time MC dose calculation responses for challenging ART problems. Costs for provisioning cloud resources are discussed in relation to performance for a publicly accessible cloud provider. We conclude that using cloud computing provides a flexible and affordable way to realise near real-time MC-based dose calculations.

2. Material and Methods

Monte Carlo simulations are known to be *embarrassingly parallel* problems. This means that little computational overhead is needed to split an MC simulation into individual tasks which can run concurrently in parallel on appropriate computational hardware. Having this property, the MC simulation is expected to scale very well with the amount of computational resources. Indeed, an excellent scaling behaviour could be demonstrated with CPU-based implementations (Tyagi et al.; 2004; Ziegenhein et al.; 2015). The parallelisation strategy is simple: An individual MC calculation is launched concurrently on every CPU core or server node. After the MC calculations are done, the resulting dose cubes are merged together by adding the deposited energy values for each voxel. The excellent scaling of this method motivates the assumption that the key to real-time MC dose calculation is to employ as many parallel resources as possible while keeping the overhead such as collecting the data from the resources as low as possible. At the same time, these resources don’t have to be physically owned but can be rented when needed from a cloud computing provider.

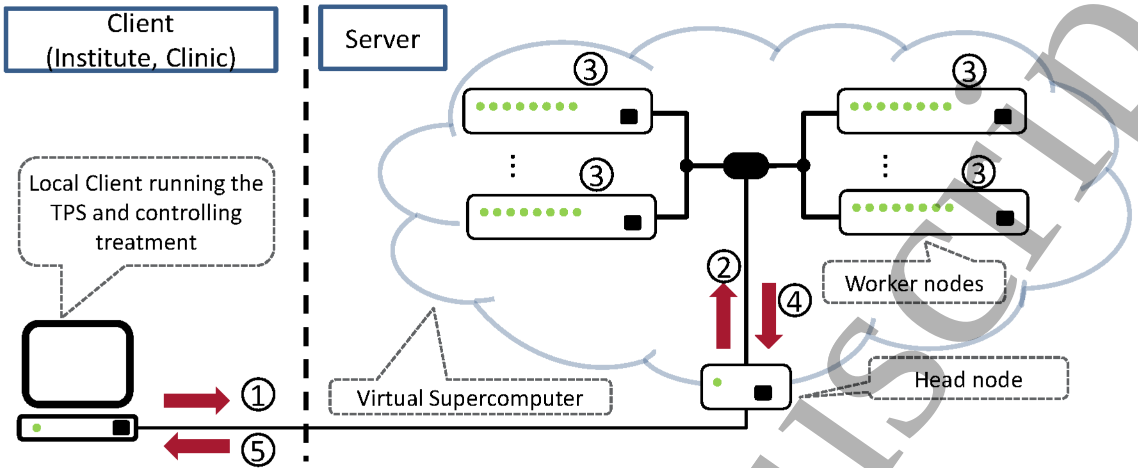


Figure 1. Workflow of the Monte Carlo based dose calculation in the cloud. A server application which performs the MC simulation is executed on the computing instances in the cloud while a client computer located in the hospital or institute provides patient data and controls the calculations.

The general workflow of our Monte Carlo dose calculation in the cloud method is shown in figure 1. The setup comprises a standard client-server architecture with the client on the left and a server with a cluster back-end on the right. The client is a local computer located in an institute or a hospital (and therefore typically behind a firewall) that runs a treatment planning system (TPS). The server side consists of a virtual cluster including a head node and several worker nodes which form a *virtual supercomputer* in the cloud. The client and the head node are connected through the Internet. The workflow starts on the client computer which requests a dose calculation during the therapy planning process. The client sends all patient specific data needed for the MC simulation to the head node of the virtual cluster (1). The head node receives the data and re-distributes it instantly to all the worker nodes (2). Each worker runs its own individual MC simulation according to the patient setup specified by the input data (3). After the simulation is completed, the workers send their results back to the head node (4). The head node merges the results and sends the final result, i.e. the merged dose distribution, back to the client computer (5).

The workflow assumes that the server application in the cloud is already running and that non-patient specific simulation data, for example, data tables holding the physical cross sections of particle interactions have already been loaded into the memory. The server application runs as long as the cloud instances are booked. A restart of the application takes a few seconds. The patient specific data which is sent to the cloud consists of one or more patient CT images, the geometrical beam setup and the intensity modulation of the beams.

Realising the complete workflow shown in figure 1 in real-time is challenging. First, the simulation itself needs to be exceptionally fast. Second, the data transport and task synchronisation overhead has to be handled significantly faster than the MC simulation itself in order to exploit the overall linear scaling of the method. The components, methods and techniques employed in our new framework in order to achieve this goal are described in detail in the following subsections.

2.1. Monte Carlo dose calculation engine

Our new cloud-based MC framework is an extension to the *PhiMC* package which we introduced in (Ziegenhein et al.; 2015). *PhiMC* adapts the physics engine from the *dose planning method* (DPM) package (Sempau et al.; 2000) for modern CPU architectures. While the original DPM code was written in Fortran, *PhiMC* was implemented in modern C++. It contains modules to simulate electrons and high energy photon transport processes which have been highly optimised for multi-core CPUs. Thread-level parallelism was implemented using OpenMP (Dagum and Menon; 1998) while an array notation was employed to make use of vectorisation on wide CPU registers. It has been demonstrated that *PhiMC* delivers accurate dose distributions by comparing to the original DPM implementation. Simulation runtimes of about 10 to 30 seconds could be achieved for clinical cases on a dual Intel Xeon workstation without compromising on resolution and accuracy.

Due to its high single node performance and accuracy, the physical engine of *PhiMC* was adapted for cloud computing with two vital changes: First, the high level parallel programming interface was changed to work with distributed computing hardware as described in section 2.3. Second, data encryption was added to protect patient data (see section 2.4). The simulation engine was embedded in a server application which is remotely controlled by a client. The communication between server and client is described in sections 2.2 and 2.3.

2.2. Data transport between client and server

One of the key features of our cloud-based MC framework is the ability to transmit data between the client and cloud very rapidly. This is important since transporting data up to the cloud and back adds a constant overhead to the simulation time. The transportation time remains the same no matter how many nodes are employed or particle histories are simulated. Thus, in order to achieve good overall performance scaling it is highly desirable to keep the transportation overhead as low as possible.

The data exchange over the internet between the client and the cluster is realised via an efficient software module which will be referred to as *data transportation unit* (DTU). An instance of this unit exists on both the client and on the server side and is able to send and receive data at the same time. The DTU consists of an ecryption/decryption module, a compression/decompression module and a Transmission Control Protocol (TCP)-based encoder and decoder. A typical workflow for transmitting data from the client to the server is illustrated in figure 2. The workflow starts with moving the data to be sent to its local DTU. Within the DTU, the data is first encrypted (1) and compressed (2), then directed to the encoder (3) which consequently transmits the

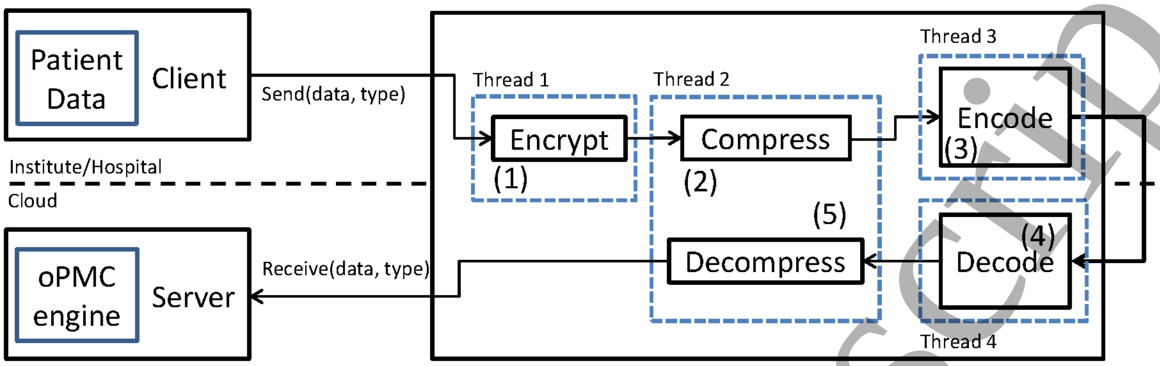


Figure 2. Structure of the data transportation unit (DTU) which is a module used by client and server to exchange patient data, results and commands securely over the internet.

data via the internet to the server side. The DTU running on the server side receives the data (4) and decodes and decompresses (5) the data before it is handed to the server application which further distributes the data to the worker nodes. The data transmission from the server back to the client is realised by the same DTU instance.

The actual data transmission is realised via the TCP protocol. Our implementation extends the *QTCPSocket* class which is part of the QT cross-platform application framework‡. The compression and decompression functionality is provided by a parallel implementation based on the Intel IPP library§. The encryption module is described in detail in section 2.4. The encoder/decoder module of the DTU as well as the encryption and compression/decompression module run in separate CPU threads which are executed concurrently. This ensures a full duplex data transportation mode which allows the DTU to receive and send data at the same time without any delays. The thread concurrency of the DTU was implemented using *QThread* objects.

To control the remote MC simulation, the DTU exploits status words. The context of the data transmitted is specified by a type identifier which is mandatory for every message. There are 4 different classes of data which are transmitted by the DTU: First, patient specific data such as the CT images and the resulting dose cube. Second, plan specific data such as beam setup and intensity modulation. Third, commands to control the MC simulation such as 'start simulation', or 'report dose distribution' and fourth, debugging and timing information that are used to identify performance hotspots. The type of the data to be transmitted also influences the flow of data through the DTU. For instance it does not make sense to compress simulation runtime informations or encrypt command words. Thus, the workflow shown in figure 2, which includes both compression and encryption, would for instance be called to transmit large patient specific data.

‡ http://www.qt.io/
§ http://software.intel.com/en-us/intel-ipp/

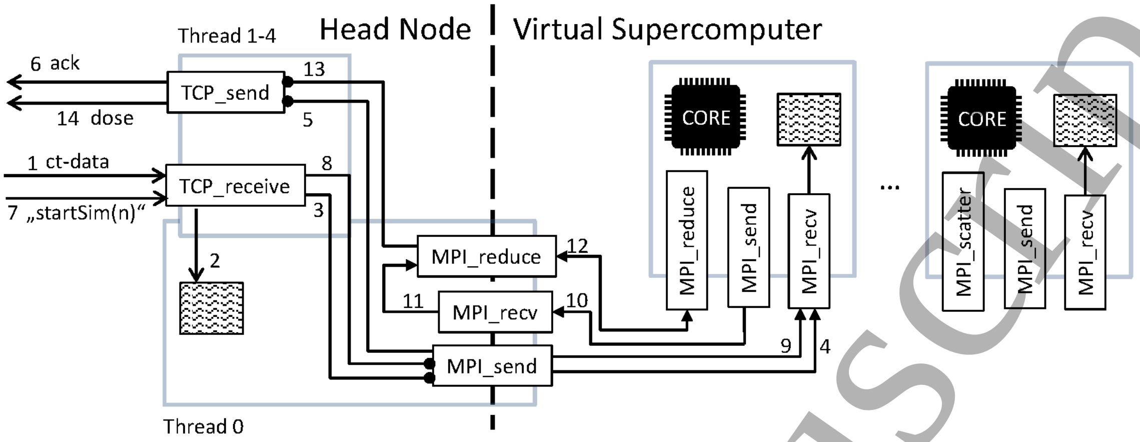


Figure 3. An example of data transport and synchronisation within the virtual supercomputer. Open arrows represent client-server communications over the internet. Communications via the signals and slots mechanism on QT are denoted by solid dots at the end of the line. Closed arrows describe communications within the cloud via MPI.

2.3. Synchronisation and parallelisation within the cloud

For our cloud implementation, the OpenMP application programming interface (API) used in *PhiMC* was replaced by the Message Passing Interface (MPI) (Gropp et al.; 1996). MPI allows to run parallel processes in distributed computing environments. Each MPI process launches an individual MC simulation on each physical CPU-core of the virtual cluster. No patient data or physics data is shared between the cores even if they reside in the same node. This is in contrast to the original implementation in *PhiMC* on shared memory systems where large data sets such as CT data and dose cubes are shared between cores. Launching completely individual MPI processes allows us to abstract the structure of the cluster: from the viewpoint of our framework the cloud is seen as one large supercomputer which provides a huge number of cores.

The server module of our MC cloud framework is responsible for an effective exchange of clinical data and control signals. It bridges the communication between the internet and the internal cloud network by utilising a number of techniques. The structure of this module is depicted in figure 3. It runs two event-loops simultaneously: A QT event loop to send and receive TCP-based data over the Internet and the MPI event-loop which communicates with the worker nodes. The QT event-loop manages an instance of the DTU which processes signals of incoming and outgoing TCP connections. The functionality of the DTU is executed in a separate dedicated thread pool (Thread 1-4 in figure 3) so that the main thread (Thread 0) is free to handle the MPI communications. Both event-loops exchange data using the asynchronous signal and slot messaging technique provided by QT. The interplay between these two event-loops is illustrated in figure 3 which shows an example of launching an MC simulation based on a new ct-data set of the patient. The communication pathway consists of the following steps:

(1) The new ct-data set is sent from the client to the head-node of the cluster. The head node receives the data and stores a copy in main memory (2). Zig-zag waves symbolise encrypted data as explained in detail in figure 4. Simultaneously a signal is sent (3) to the main thread to report that new data is ready to be distributed to the worker. The MPI thread sends the new CT data to all processes (cores) of the virtual cluster (4) which all store an individual copy of the new data in memory. The send method is triggered via a blocking function call and returns as soon as the data is delivered. The delivery of the data is signaled to the DTU (5) and acknowledged by sending a status word (ack) back to the client (6). The client can now trigger the simulation by sending a *startSim* command to the head-node (7). Note that this command has a parameter n which denotes the number of histories to be simulated. The *startSim* command is signaled to the main thread (8) which passes it along to the simulation processes (9). Each core performs the simulation and reports back to the head-node when finished (10). The head-node then triggers an MPI operation (11) which requests all dose cubes from the worker and automatically adds the energy contribution of all cubes together (12). The main thread signals to the DTU that the result is ready (13) for download to the client computer (14). Arrows 4, 9, 10 and 12 reach out to all cores in the virtual supercomputer simultaneously. For the sake of clarity only the connection to one core is shown in the graph.

The example in figure 3 shows how the different parts of the server module interact. Similar workflows are defined for setting up the beam configurations and fluence modulation, configuring the simulation physics of the worker and collecting debugging and timing data.

2.4. Cloud methodology, security and reliability

Cloud computing has become an important economical driver for many large and small businesses in the past years. It allows tapping into required IT resources and applications using pay-as-you-go models or long lease options. Resources can be used on-demand according to the actual needs of the users and released when no longer needed. Furthermore Amazon (Seattle, WA, USA) and other cloud providers implement rigorous security practices on their side which satisfy the requirements and standards of governments and security agencies. Ultimately the level of security used in the cloud is the responsibility of the user and multiple levels of security can be implemented. For instance, the communication channel to the cloud can be encrypted, the data itself can be encrypted or both at the same time. The point at which the data is decrypted can also vary from the entry point to the cloud, to the cloud server itself, or to the processor. For our tests, we selected Amazon Web Services (AWS) as it provides a mature platform. The Amazon Elastic Compute Cloud (EC2) products on which the results of this work were created can be used to architect applications in alignment with

HIPAA ‖ and HITECH ¶ compliance requirements.

AWS has a special service called Virtual Private Cloud (VPC) which enables the creation of an isolated subnet including selection of private IP addresses, so that compute servers existing in it are not visible from the Internet. The communication with the external world (the Internet) is carried through a designated head node which is configured to have a public IP address. Additionally the VPC service includes security groups and network access control lists, which were used to further restrict the communication with the head node to two specific ports and a restricted set of client Internet addresses so that only desired client(s) can access the service.

Cloud providers nowadays offer various tools and APIs to automate creation and manipulation of instances in the cloud. Once we fashioned a desired virtual image preloaded with all required libraries and tools, we could start a required number of instances programmatically from the client workstation and shut them down when the tests were finished. Thus it is entirely possible to create a client which not only schedules a task and receives an answer but also orchestrates the cloud instances automatically. Since our tests were taking only seconds whereas it takes a few minutes for an instance to come up, we usually provisioned a maximum number of instances before running a series of tests.

In order to realise a very fast MC dose calculation, the number of computational nodes leased from a cloud service provider has to be chosen with care. For example, preliminary tests can be run to estimate what sample size and sort of instances are needed to finish computation in a given time limit. Then at production time, the client can make a rational choice by spawning the right number of instances and therefore minimising the cost. Finally, reliability of calculations in the sense of fault tolerance can be increased by soliciting redundant calculations from other cloud zones or regions. Doing so will obviously increase the running cost, and should be considered as an operational decision. Multiple scenarios are possible here including oversampling, i.e. collecting and incorporating all redundant calculations rather than disregarding them and therefore increasing the accuracy of MC simulations.

Privacy and security are extremely important when it comes to clinical applications of a cloud-based Monte Carlo dose calculation. In the following we describe a powerful encryption module which implements a processor to processor encryption model and is part of the DTU as described in section 2.2. This method and module can be used in addition to other security methods described earlier. The encryption module ensures that all patient specific data is held encrypted when transmitted or handled on third party hardware. The decryption of patient specific data (e.g. CT data) is performed on-the-fly during the simulation only. Figure 4 illustrates the working principle. The workflow shows the handling of patient specific data on the example of the CT image. The CT data is illustrated as a 2D grid plane in the TPS computer. All sensitive

‖ U.S. Health Insurance Portability and Accountability Act: http://www.hhs.gov/ocr/privacy/

¶ Health Information Technology for Economic and Clinical Health Act: http://www.hhs.gov/ocr/privacy/hipaa/administrative/enforcementrule/hitechenforcementifr.html

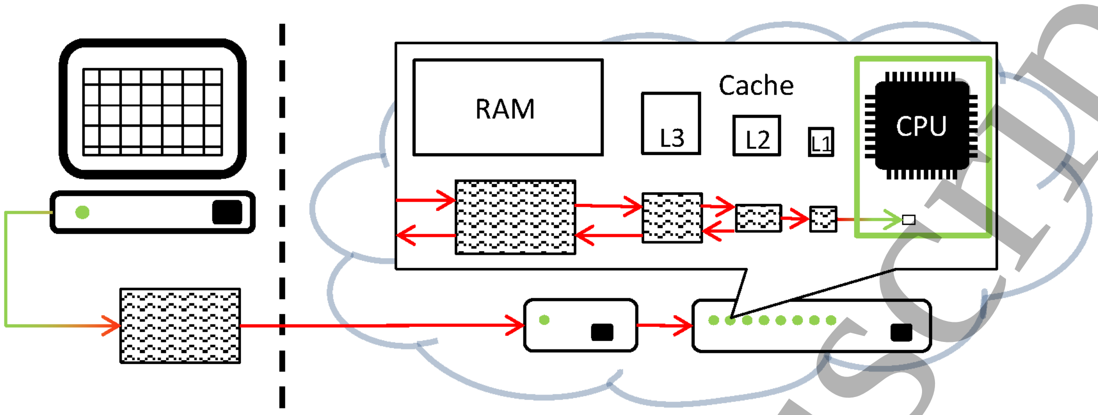


Figure 4. Encryption/Decryption workflow of patient-specific data. Patient-specific data is securely encrypted (arrow from green to red) before it leaves the client and decrypted only in the registers of the CPU for immediate use. Patient specific data residing in RAM and various levels of cache memory (L3,L2,L1) is always encrypted.

data is encrypted on the client side before it is sent over to the cloud. Encrypted data is symbolised by zig-zag waves hiding the ct-data. The cloud head node receives the encrypted data and passes it along to the worker nodes. On the worker nodes the encrypted data is stored in main memory. When requested by the simulation physics the corresponding data locality is fetched by the cache levels and tansported into the CPU registers. At this stage the data is still encrypted. Only in the CPU registers the requested data word is decrypted temporally and directly used in the respective arithmetic operations. The data word is encrypted again before it leaves the CPU and gets evicted to the main memory. Thus, plain patient data only exists temporally and in small quantities in the CPU registers while it is held encrypted in all levels of memory on the worker nodes.

The encryption is done by using the Advanced Encryption Standard (AES-128). AES (Daemen and Rijmen; 1999) is a symmetric block cipher which was adopted in 2002 by the U.S. government to protect classified information (NIST-FIPS; 2001). It is the first (and only) publicly accessible standard approved by the National Security Agency (NSA) for top secret information. Due to its popularity it is widely used for many applications in every-day life. The chipper algorithm itself is efficiently supported in hardware via the AES New Instruction set (AES-NI) on all modern processors[+]. Using AES-NI, encrypting and decrypting data can be performed directly in the CPU registers by calling a few assembler instructions. The hardware implementation of AES guarantees high performance and the correctness of the algorithm. A compiler intrinsics based AES-NI implementation was developed and is used in the DTU (see figure 2) as well as for handling encrypted data in the MC simulation physics.

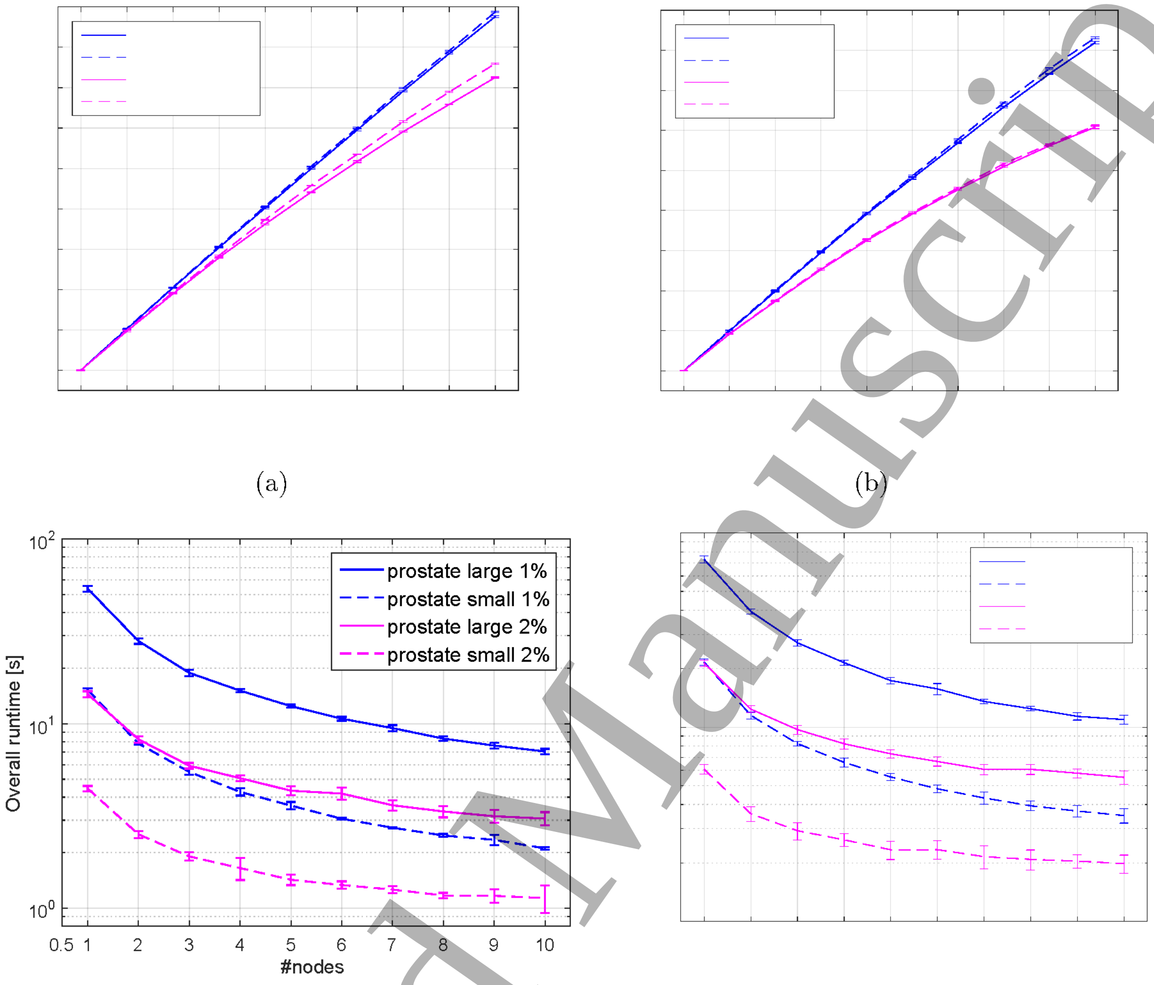
[+] http://ark.intel.com/search/advanced/?s=t&AESTech=true

Case/ Size	Prostate 1%	 2%	Liver 1%	 2%
Original $(3\ mm)^2 \times 3\ mm$	105m	28m	80m	19m
Large $(1.5\ mm)^2 \times 3\ mm$	210m	53m	250m	68m

Table 1. Number of particle histories to be simulated to achieve 1% and 2% statistical uncertainty on the large and original sized prostate and liver treatment cases.

2.5. Test data

The performance of our MC cloud framework is tested against patients from the publicly accessible CORT patient dataset (Craft et al.; 2014). From the shared data set* we picked the prostate case and the SBRT liver case. Both cases were planned with step-and-shoot IMRT using 5 photon beams for prostate and 7 photon beams for liver with the intensity modulation as provided in the CORT data set. The beamlet size is $(1\ cm)^2$ for both cases while the prostate target volume comprises 256.2 cm^3 and the liver comprises 156.5 cm^3. The resolution of the planning CT-cube is 3 mm in each dimension for both patients. The size of the ct-cube is $184 \times 184 \times 90$ voxels for prostate and $217 \times 217 \times 168$ voxels for the liver patient. In order to demonstrate that our method is also capable of dealing with higher resolutions we interpolated the original ct-data set to $368 \times 368 \times 90$ voxels for prostate and $434 \times 434 \times 168$ voxels for liver. This results in a resolution of $(1.5\ mm)^2 \times 3\ mm$ for both patient images. The number of CT-slices as well as the resolution of the fluence matrix remains unchanged. The interpolated patient data will be referred to as *large* while the original data set will be referred to as *original*, hereinafter. The dose distribution of both patients on both image sizes is simulated up to 1% and 2% statistical uncertainty. Thus, 8 MC simulation problems are studied in total. The number of particle histories simulated for each problem can be found in table 1. Tests were performed on AWS using *c4.8xlarge* compute-optimised instances which allowed the highest network performance comparable with 10 Gbps Ethernet. These instances feature Intel Haswell processors which were the best processors available at the time. Each instance was used to launch 18 MPI processes. Although the instances are typically cheaper in the US, in order to minimise the latency to the cloud, we provisioned our instances in Ireland at the typical cost of $1.811 per hour. All runtimes have been collected on a working day (Thursday) between 11am and 7pm using a shared Internet link of our institute.



(a)

(b)

(c)

(d)

Figure 5. (a),(b): Performance scaling of the MC simulation within the virtual supercomputer in the cloud. (c),(d): Overalll dose calculation runtime in the cloud

3. Results

The performance of our cloud-based dose calculation framework is analysed in figure 5. Figure 5(a) and 5(b) show the performance scaling of Monte Carlo simulations within the cloud for the selected dose calculation scenarios. The scaling data was measured over the simulation process only while data transport times to and from the cloud are not included. The scaling however takes into account the distribution of the clinical data to the worker in the cloud, the MC simulation itself and the superposition of all resulting dose cubes from the worker (i.e. processes 2-4 of the workflow in figure 1). Figure 5(c) and figure 5(d) show the overall runtime of the complete dose calculation workflow which includes the MC simulation, data transportation and synchronisation processes to, from and within the cloud and all other overhead. The overall runtimes

* http://dx.doi.org/10.5524/100110

	Size	Stat.	1 node	10 nodes		
		Uncertainty	Simulation	Simulation	Overhead	Overall
Prostate	Large	1%	50.03 ± 0.28	5.13 ± 0.01	1.94 ± 0.24	7.07 ± 0.24
		2%	12.77 ± 0.11	1.55 ± 0.00	1.51 ± 0.25	3.06 ± 0.25
	Original	1%	14.40 ± 0.05	1.46 ± 0.00	0.65 ± 0.03	2.11 ± 0.03
		2%	3.87 ± 0.01	0.450 ± 0.00	0.69 ± 0.19	1.14 ± 0.19
Liver	Large	1%	67.90 ± 0.52	7.39 ± 0.02	3.56 ± 0.59	10.95 ± 0.60
		2%	18.73 ± 0.13	2.64 ± 0.01	2.87 ± 0.45	5.52 ± 0.45
	Original	1%	19.70 ± 0.12	2.12 ± 0.00	1.39 ± 0.30	3.51 ± 0.30
		2%	4.77 ± 0.03	0.67 ± 0.00	1.31 ± 0.21	1.98 ± 0.21

Table 2. MC simulation runtimes in seconds measured on our cloud-based Monte Carlo dose calculation framework employing only 1 workstation (18 physical CPU cores) locally and 10 nodes (180 CPU cores) in the cloud. Standard deviations are calculated from at least 10, typically 20-25 repetitions per test. Standard deviations below 10 milliseconds are printed as 0.00

can be understood as *wall-clock* times measuring the actual time span as perceived by the user between triggering the dose calculation on the client computer and receiving the resulting dose cube back from the cloud. The overall and simulation runtimes on 10 worker nodes as well as the simulation time on one node are listed in table 2. The overhead in this table corresponds to the overall runtime of simulating one batch (1000 particle histories) per CPU core. The actual simulation time of 1000 histories is in the order of 5 ms and therefore negligible. Thus, the overhead column in table 2 denotes the wall-clock time for all other processes including data transportation, synchronisation, scheduling, network latency, internal data handling processes etc. - everything except the actual physical particle track simulation.

Table 3 analyses runtime (T) and bandwidth (BW) of selected vital overhead processes for the example of the prostate patient data set. The columns of the table denote the following: *ENC*, encrypting the CT data in the DTU (see (2) in figure 2); *COM*, compressing the patient image (see (1) in figure 2); *UP*, transferring the encrypted and compressed CT data to the head node of the cloud; *MPI*, merging the results of the worker nodes and *DOWN*, sending the merged dose cube back to the client. The on-the-fly decryption of the patient data during the simulation as introduced in section 2.4 accounts for a performance loss of about 6% relative to the simulation time. The term bandwidth is defined as the amount of processed data in a fixed time. More specifically, bandwidths in table 3 are measured in *megabits per second* (Mbps) so that the notion of bandwidth is: transferred or processed million bits of data in one second.

			CT cube			Dose cube	
			ENC	COM	UP	MPI	DOWN
Prostate	Original	T	22.8	37.4	35.2	78.2	120.4
		BW	2468.6	1371.9	167.2	2627.1	406.7
	Large	T	23.4	134.9	320.2	319.1	245.0
		BW	9525.2	1125.4	151.3	2564.6	805.0

Table 3. Runtime (T) in milliseconds and bandwidth (BW) in Mbps for encrypting (ENY), compressing (COM) and transmitting (UP) the CT data set into the cloud as well as merging the dose cubes from all workers (MPI) and sending the result back to the client (DOWN). Errors are small and not listed for the sake of clarity.

4. Discussion

In this work, we introduced a highly integrated cloud-based framework which provides near real-time Monte Carlo dose calculations for high energy photons. The framework makes use of the embarrassingly parallel nature of the MC simulation technique by employing a high number of CPU cores of a virtual supercomputer in the cloud while reducing the overhead as far as possible. The performance analysed in figure 5 proves two main results of this work. First, the linear scaling of the MC simulations which was observed on shared-memory systems and CPU clusters (Ziegenhein et al.; 2015) can be reproduced in the cloud. Second, the overhead of the MC simulation, which consists of transferring data between the client and the cloud, scheduling the processes and synchronising the workflow, can be limited to only a few seconds for patient data sizes relevant for clinical applications. This allows us to leverage the excellent scaling of MC simulations in the cloud. The achieved dose calculation times can be considered to fulfill near real-time requirements.

Figure 5(a) and 5(b) analyse the performance of the MC simulation within the virtual supercomputer in the cloud. The graphs show an almost perfect linear scaling up to 10 nodes for the simulations achieving 1% statistical uncertainty. The scaling of the 2% uncertainty cases, which simulate significantly less particles histories, deteriorates when using more than 5 nodes. This is caused by the overhead which falls in the same order of magnitude as the actual simulation time for 2% uncertainty cases. It can be concluded that the computational effort of simulating the dose distribution up to only 2% uncertainty is too small to fully exploit the resources of more than 5 nodes using the current simulation physics and patient data resolution.

The total overhead which is due to transporting data between client and cloud, synchronising processes within the cloud as well as compressing and encrypting patient specific data is printed in table 2. It depends on the size of the patient data set. The liver patient CT and dose cube are significantly larger than the cubes corresponding to the prostate patient. Similarly, the *large* configuration of each patient has 4 times more voxels than the *original* patient data set, which accounts for a higher overhead.

We observed a slightly higher overhead for the 1% uncertainty simulation. This is due to the fact that the resulting dose distribution is more complex when more particles are simulated leading to less efficient data compression and thus higher data volumes to be transmitted back to the client. The overhead does not dependent on the number of CPUs employed for the simulation. Table 3 reveals some of the most interesting aspects of the data transportation overhead. It shows that encrypting, compressing and transmitting a complete patient data set in clinical resolution can be done in real-time on modern hardware. Please note that table 3 only lists selected aspects of the total overhead which is printed in table 2.

The MPI based parallelisation of the simulation algorithm generalises the physical structure of the worker nodes. It does not depend on the number of cores in a node or where the nodes are located. It is even possible that part of cloud nodes are physically located on a different continent (however, this would be ill-adviced in regard to networking performance). Due to this flexibility our framework also runs on a local cluster or on a remote single node server.

So far we described the functionality of our newly developed framework. How does computing in the cloud compare to computing on-premise (using computer hardware locally) in terms of performance and costs for MC dose calculation? On-premise one could use a workstation at the office desk or offload the MC simulation to a local cluster. The simulation time for running the MC calculations on one workstation (node) locally, without any cloud or network interfaced involved is printed in table 2 in the third column. The simulation starts after all data was loaded into the dose calculation application and ends with the resulting dose cube being present in memory. These are the same start and end points as assumed for the overall cloud-based performance measurements listed in the last column of table 2. Thus, the runtime in both columns can be compared directly. The runtimes show that on one node all simulation experiments are performed in less than one minute. Faster runtimes can be achieved on-premise by employing a local cluster for the dose calculation. However, even a cluster in close proximity to the TPS client needs to be connected via a network to the planning computer which builds overhead similar to the one discussed for our cloud solution. If the cluster is located within a trusted area one could consider dropping data encryption. However, this would hardly change the runtime since due to compression, transmittion, synchronisation and distribution of data the workflow would be exactly the same. Even the magnitude of the overhead would be comparable as we could show in table 2. The respective cloud overhead was measured at the Institute of Cancer Research, which provides a stable 10 Gbps Internet link. This network rate is quite fast even for local network infrastructures. Most clinics and institutes usually may not provide more than 1 Gbps internal network bandwidth. Apart from minor latency differences we can assume that the overhead of using a local cluster is comparable to the overhead we have measured using the cloud configuration.

However, a 10 Gbps Internet fiber is not necessary in order to benefit from scalable cloud-based MC simulations. Table 3 shows that our framework only uses about 160

Mbps to transport patient data to the cloud and up to 800 Mbps to receive results. These numbers show that a 1 Gbps Internet link may be enough to achieve the performance stated in this paper. In case there is only a 100 Mbps Internet link present the additional overhead would still allow a (near) real-time scaling as a simple calculation demonstrates: For the prostate case the *UP* transfer would take about 24 ms and 164 ms longer for the original and large prostate case while the downstream of the dose results would be delayed by 361 ms and 1.7s, respectively. Engineering efficient data transport on 10 Gbps for relatively small data such as used for MC dose calculation is challenging and we decided to settle with the bandwidths printed in table 3. On larger data sets we achieved 3-4 Gbps network performance to and from the Amazon cloud even on peak day times which leaves some room to improve our framework in the future.

Provisioning one *c4.8xlarge* instance (18 CPU cores) from Amazon costs $1.811 per hour in Europe and $1.591 per hour in the US including energy, maintenance and network. The consumer hardware price of one server similar to the Amazon node instance is about $7k♯ which is equal to renting the Amazon node for 4400 hours. Assuming a 35 hour use per week the hardware cost of one server will amortize in approximately 2 years and 5 months. This calculation takes into account that cloud node instances can be easily terminated during night-time and on weekends when they are not needed. This can be done automatically using a script and takes less than one minute. A virtual cluster in the cloud can be built by provisioning a number of instances. The cost increases linearly with the number of resources, e.g. for a 10 node cloud cluster 10 × $1.591 per hour has to be payed. Building a cluster on-premise costs more than the nodes it contains. Additional expenses need to be planned for housing, cooling, networking infrastructure, energy and maintenance. We estimate that hardware costs in the order of $100k are required for a cluster which is comparable to the cloud configuration used to achieve the results presented in this paper. This estimation includes the network infrastructure and a separate login node. For the assumed on-premise hardware costs a similar 10 node cluster in the cloud can be provisioned for 6285 hours or approximately 3 years and 5 months assuming again a 35 hours use per week. The cost estimations drawn in this paragraph do not include VAT and potential volume discounts on both sides. The on-premise calculation does not include cost of energy, maintenance, housing facilities, cooling and redundancy arrangement. These numbers are difficult to estimate and depend on individual factors and local conditions. Omitting these costs, which can be significant, puts an advantage to on-premise cluster solutions in this discussion and draws a pessimistic lower bound for the cost efficiency of the cloud solution. In addition to that it can be argued that prices in the cloud drop over the assumed time span of 2-3 years due to technological advances according to Moore's Law.

The cost estimation demonstrates that using cloud computing is an affordable alternative to on-premise computing for MC dose calculations. Especially a research environment can benefit from provisioning resources temporarily to realise scientific

♯ assuming a Dell PowerEdge R430 2xE5-2630v3 and 64 GB RAM configured on www.dell.com projects without upfront investment costs for expensive hardware. The elastic nature of cloud computing allows to provision the right type and size of the computing resources and adapt it dynamically to changing needs, almost instantly. If just fast dose calculations are needed one node would be enough. More nodes can be dynamically added to the virtual cluster to reduce calculation time even further in case real-time requirements need to be satisfied. (Near) Real-time MC simulations are required for modern ART scenarios (Jia et al.; 2015). We are currently using the presented framework in on-line adaptive re-planning studies and plan to upgrade it soon for treatment planning on MR-Linac machines.

MC simulations in the cloud can also be used to support every-day clinical treatment planning. We showed that data can be transferred securely while peak workloads can be handled by allocating elastic resources. For clinical use, throughput would probably be more important than real-time calculation speeds which is best achieved by provisioning multiple instances each providing an individual MC server application in order to simulate multiple patient doses at a time. The main risk in this scenario would be to maintain a stable Internet connection to the cloud. The risk of hardware failures is the responsibility of the cloud provider. In case of a malfunction the image of a running node is switched to another instance automatically. A general cost comparison between a cloud-based MC dose calculation service and on-premise dose calculations for clinical use cannot be drawn in the scope of this paper. Individual requirements and circumstances need to be taken into consideration for that.

5. Conclusion

Cloud computing enables affordable (near) real-time Monte Carlo dose calculations for everybody, everywhere. It provides a means to access much more computational resources than usually available on-premise in an institute or a clinic. Monte Carlo simulations are especially well suited to scale out in a cloud environment which creates a huge potential to accomplish accurate dose calculations for adaptive treatment planning scenarios with (near) real-time requirements.

6. Acknowledgments

This research at The Institute of Cancer Research was supported by Cancer Research UK under Programme C33589/A19727 and NHS funding to the NIHR Biomedical Research Centre at The Royal Marsden and The Institute of Cancer Research.

References

Craft, D., Bangert, M., Long, T., Papp, D. and Unkelbach, J. (2014). Shared data for intensity modulated radiation therapy (imrt) optimization research: the cort dataset, *GigaScience* **3**(1): 1.

Daemen, J. and Rijmen, V. (1999). Aes proposal: Rijndael.

Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming, *IEEE computational science and engineering* **5**(1): 46–55.

Gropp, W., Lusk, E., Doss, N. and Skjellum, A. (1996). A high-performance, portable implementation of the mpi message passing interface standard, *Parallel computing* **22**(6): 789–828.

Hissoiny, S., Raaijmakers, A., Ozell, B., Després, P. and Raaymakers, B. (2011). Fast dose calculation in magnetic fields with gpumcd, *Physics in medicine and biology* **56**(16): 5119.

Jahnke, L., Fleckenstein, J., Wenz, F. and Hesser, J. (2012). Gmc: a gpu implementation of a monte carlo dose calculation based on geant4, *Physics in medicine and biology* **57**(5): 1217.

Jia, X., Gu, X., Graves, Y. J., Folkerts, M. and Jiang, S. B. (2011). Gpu-based fast monte carlo simulation for radiotherapy dose calculation, *Physics in Medicine and Biology* **56**(22): 7017.

Jia, X., Schümann, J., Paganetti, H. and Jiang, S. B. (2012). Gpu-based fast monte carlo dose calculation for proton therapy, *Physics in medicine and biology* **57**(23): 7783.

Jia, X., Xu, X. G., Orton, C. G. et al. (2015). Gpu technology is the hope for near real-time monte carlo dose calculations, *Medical physics* **42**(4): 1474–1476.

Lagendijk, J. J., Raaymakers, B. W., Raaijmakers, A. J., Overweg, J., Brown, K. J., Kerkhof, E. M., van der Put, R. W., Hardemark, B., van Vulpen, M. and van der Heide, U. A. (2008). Mri/linac integration, *Radiotherapy and Oncology* **86**(1): 25–29.

Lee, V. W., Kim, C., Chhugani, J., Deisher, M., Kim, D., Nguyen, A. D., Satish, N., Smelyanskiy, M., Chennupaty, S., Hammarlund, P. et al. (2010). Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu, *ACM SIGARCH Computer Architecture News*, Vol. 38, pp. 451–460.

Miras, H., Jiménez, R., Miras, C. and Gomà, C. (2013). Cloudmc: a cloud computing application for monte carlo simulation, *Physics in medicine and biology* **58**(8): N125.

NIST-FIPS (2001). Announcing the advanced encryption standard (aes), *Federal Information Processing Standards Publication* **197**: 1–51.

Pratx, G. and Xing, L. (2011). Monte carlo simulation of photon migration in a cloud computing environment with mapreduce., *Journal of biomedical optics* **16**(12): 125003.

Sempau, J., Wilderman, S. J. and Bielajew, A. F. (2000). Dpm, a fast, accurate monte carlo code optimized for photon and electron radiotherapy treatment planning dose calculations, *Physics in medicine and biology* **45**(8): 2263.

Su, L., Yang, Y., Bednarz, B., Sterpin, E., Du, X., Liu, T., Ji, W. and Xu, X. G. (2014). Archerrt–a gpu-based and photon-electron coupled monte carlo dose computing engine for radiation therapy: Software development and application to helical tomotherapy, *Medical physics* **41**(7): 071709.

Townson, R. W., Jia, X., Tian, Z., Graves, Y. J., Zavgorodni, S. and Jiang, S. B. (2013). Gpu-based monte carlo radiotherapy dose calculation using phase-space sources, *Physics in medicine and biology* **58**(12): 4341.

Tyagi, N., Bose, A. and Chetty, I. J. (2004). Implementation of the dpm monte carlo code on a parallel architecture for treatment planning applications, *Medical physics* **31**(9): 2721–2725.

Ziegenhein, P., Pirner, S., Kamerling, C. P. and Oelfke, U. (2015). Fast cpu-based monte carlo simulation for radiotherapy dose calculation, *Physics in medicine and biology* **60**(15): 6097.