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, SM2 5NG, United Kingdom

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

Received 22 August 2016, revised 11 January 2017
Accepted for publication 31 January 2017
Published 5 May 2017

Abstract
Near real-time application of Monte Carlo (MC) dose calculation in clinic and research is hindered by the 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 (CPUs). 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 cost 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. Advanced encryption standards were used to add an additional 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 seconds to 10.9 seconds for simulating a clinical prostate and liver case up to 1% statistical uncertainty. The computation runtimes include the transportation of data 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.

Keywords: cloud computing, Monte Carlo, MPI, HPC, real-time

(Some figures may appear in colour only in the online journal)

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 recent years, various groups have developed fast MC dose calculation solutions on different computational platforms. In particular, the utilisation of graphics processing units (GPUs) in an attempt to reduce calculation times significantly has been very popular. A number of GPU-based implementations have been introduced for photons, electrons and protons, for example Hissoiny et al (2011), Jia et al (2011), Jahnke et al (2012), Jia et al (2012) and 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 (Tyagi et al 2004, Su et al 2014, 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 in 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 quantity 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 quantity 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 is currently no cloud-based MC solution 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 do not have to be physically owned, but can be rented when needed from a cloud computing provider.

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 to achieve this goal are described in detail in the following sections.

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 that 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 by comparison to the original DPM implementation that PhiMC delivers accurate dose distributions. Simulation runtimes of about 10 to 30 s 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 that is controlled remotely 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 because 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 the 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 encryption/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 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 effected by the same DTU instance.

The actual transmission of data 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

1 www.qt.io/
2 http://software.intel.com/en-us/intel-ipp/
ensures a full duplex data transportation mode, allowing 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. Four different classes of data 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 is 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 information or encrypt command words. Thus, the workflow shown in figure 2, which includes both compression and encryption, would typically be called to transmit large patient-specific data sets.

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 running 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 providing a huge number of cores.

The server module of our MC cloud framework is responsible for the effective exchange of clinical data and control signals. It bridges the communication between the Internet and the internal cloud network using 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 \textit{startSim} command to the head-node (7). Note that this command has a parameter \( n \) denoting the number of histories to be simulated. The \textit{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 recent 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\(^3\) and HITECH\(^4\) 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 have been 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 programmatic 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 MC 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 in the

---

\(^3\)
US health insurance portability and accountability act: [www.hhs.gov/ocr/privacy/](http://www.hhs.gov/ocr/privacy/)

\(^4\)
Health information technology for economic and clinical health act: [www.hhs.gov/ocr/privacy/hipaa/administrative/enforcementrule/hitechenforcementrif.html](http://www.hhs.gov/ocr/privacy/hipaa/administrative/enforcementrule/hitechenforcementrif.html)
example of a CT image. The CT data is illustrated as a 2D grid plane in the TPS computer. All sensitive 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. At 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 transported into the CPU registers. At this stage the data is still encrypted. Only in the CPU registers is the requested data word decrypted temporarily and directly used in the respective arithmetic operations. The data word is encrypted again before it leaves the CPU and is passed to the main memory. Thus, plain patient data only exists temporarily and in small quantities in the CPU registers, while it is held encrypted at 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, adopted in 2002 by the US 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 everyday 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.

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 five photon beams for prostate and seven photon beams for liver with the intensity modulation as provided in the CORT data set. The beamlet size is (1 cm²) for both cases while the prostate target volume comprises 256.2 cm³ and the liver comprises 156.5 cm³. The resolution of the planning CT-cube is 3 mm in each dimension for both patients. The size of the CT-cube is 184 × 184 × 90 voxels for prostate and 217 × 217 × 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 × 368 × 90 voxels for prostate and 434 × 434 × 168 voxels for liver. This results in a resolution of (1.5 mm)² × 3 mm for both patient images. The number of CT-slices and the resolution of the fluence matrix remain 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, eight 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

5 http://ark.intel.com/search/advanced/?s=t&AESTech=true
6 http://dx.doi.org/10.5524/100110
the typical cost of $1.811 per hour. All runtimes were collected on a working day (Thursday) between 11am and 7pm using a shared Internet link of our institute.

3. Results

The performance of our cloud-based dose calculation framework is analysed in figure 5. Figures 5(a) and (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).

Figures 5(c) and (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 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 ten 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 data processed in a fixed time. More specifically, bandwidths in table 3 are measured in megabits per second (Mbps) so that the notion of bandwidth is: million bits of data transferred or processed in one second.
Figure 5. (a), (b): performance scaling of the MC simulation within the virtual supercomputer in the cloud. (c), (d): overall dose calculation runtime in the cloud.

Table 2. MC simulation runtimes in seconds measured on our cloud-based Monte Carlo dose calculation framework employing only one workstation (18 physical CPU cores) locally and ten 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 ms are printed as 0.00.

| Size     | Stat. uncertainty | 1 node | 10 nodes |
|----------|-------------------|--------|----------|
|          |                   | Simulation | Overall | Simulation | Overhead | Overall   |
|          |                   | 1 node | Simulation | Overhead | Overall |
| Prostate | 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    | 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 |
4. Discussion

In this work, we introduced a highly integrated cloud-based framework which provides near real-time MC 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 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 to clinical applications. This allows us to take advantage of the excellent scaling of MC simulations in the cloud. The dose calculation times achieved can be considered to fulfill near real-time requirements.

Figures 5(a) and (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 ten nodes for the simulations achieving 1% statistical uncertainty. The scaling of the 2% uncertainty cases, which simulate significantly fewer particle histories, deteriorates when using more than five 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 five nodes using the current simulation physics and patient data resolution.

The total overhead due to transporting data between client and cloud, synchronising processes within the cloud and 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 four 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.

|                      | 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    |
| Prostate 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.
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 some cloud nodes are physically located on a different continent (however, this would be ill advised in regard to networking performance). Due to this flexibility, our framework can also run on a local cluster or on a remote single node server.

So far we have described the functionality of our newly developed framework. How does computing in the cloud compare to computing on-premises (using computer hardware locally) in terms of performance and costs for MC dose calculation? On-premises 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 interface involved is printed in table 2 in the third column. The simulation starts after all data is 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 that 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, transmission, synchronisation and distribution of data the workflow would be exactly the same. Even the magnitude of the overhead would be comparable, as can be seen from 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. 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 1.7 s 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 at 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 $7 k, which is equal to renting the Amazon node for 4400h. Assuming a 35 h use per week the hardware cost of one server will amortize in approximately two years and five months. This calculation takes into account that cloud node instances can be easily terminated during night-time.

---

7 assuming a Dell PowerEdge R430 2xE5-2630v3 and 64 GB RAM configured on www.dell.com
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 quantity of resources, e.g. for a ten node cloud cluster $10 \times \$1.591$ 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 $100\,\text{k}$ 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 ten node cluster in the cloud can be provisioned for 6285 h or approximately three years and five months, assuming again a 35 h use per week. The cost estimations drawn in this paragraph do not include VAT or potential volume discounts on both sides. The on-premise calculation does not include cost of energy, maintenance, housing facilities, cooling and redundancy arrangements. These numbers are difficult to estimate, and depend on individual factors and local conditions. Omitting these costs, which can be significant, puts an advantage with on-premise cluster solutions in this discussion and draws a pessimistic lower bound for the cost efficiency of the cloud solution. In addition to this, 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. A research environment in particular can benefit from provisioning resources temporarily to realise scientific projects without upfront investment costs for expensive hardware. The elastic nature of cloud computing allows provisioning of the right type and size of computing resources and adapting them dynamically to changing needs, almost instantly. If just fast dose calculations are needed, one node will 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 have 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, and 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 concern the ability 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 would require individual requirements and circumstances to be taken into consideration, and so cannot be drawn in the scope of this paper.

5. Conclusion

Cloud computing enables affordable (near) real-time Monte Carlo dose calculations for everybody, everywhere. It provides a means to access much greater computational resources than are usually available on-premises 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.

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 2014 3:37
Daemen J and Rijmen V 1999 Aes proposal: Rijndael The Rijndael Block Cipher
Dagum L and Menon R 1998 Openmp: an industry standard api for shared-memory programming IEEE Comput. Sci. Eng. 5 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 Comput. 22 789–828
Hissoiny S, Raaijmakers A, Ozell B, Després P and Raaymakers B 2011 Fast dose calculation in magnetic fields with gpumcd Phys. Med. Biol. 56 5119
Jahneke L, Fleckenstein J, Wenz F and Hesser J 2012 Gmc: a gpu implementation of a Monte Carlo dose calculation based on geant4 Phys. Med. Biol. 57 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 Phys. Med. Biol. 56 7017
Jia X, Schümann J, Paganetti H and Jiang S B 2012 Gpu-based fast Monte Carlo dose calculation for proton therapy Phys. Med. Biol. 57 7783
Jia X et al 2015 Gpu technology is the hope for near-real-time Monte Carlo dose calculations Med. Phys. 42 1474–6
Lagendijk J J, Raaijmakers 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 Radiother. Oncol. 86 25–9
Lee V W et al 2010 Debunking the 100x gpu versus cpu myth: an evaluation of throughput computing on cpu and gpu ACM SIGARCH Computer Architecture News vol 38 (New York: ACM) pp 451–60
Miras H, Jiménez R, Miras C and Gomà C 2013 Cloudmc: a cloud computing application for Monte Carlo simulation Phys. Med. Biol. 58 N125
NIST-FIPS 2001 Announcing the advanced encryption standard (aes) Federal Information Processing Standards Publication vol 197 (Washington, DC: US Department of Commerce) pp 1–51
Prax G and Xing L 2011 Monte Carlo simulation of photon migration in a cloud computing environment with mapreduce J. Med. Opt. 16 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 Phys. Med. Biol. 45 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 Med. Phys. 41 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 Phys. Med. Biol. 58 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 Med. Phys. 31 2721–5
Ziegenhein P, Pirner S, Kamerling C P and Oelke U 2015 Fast cpu-based Monte Carlo simulation for radiotherapy dose calculation Phys. Med. Biol. 60 6097