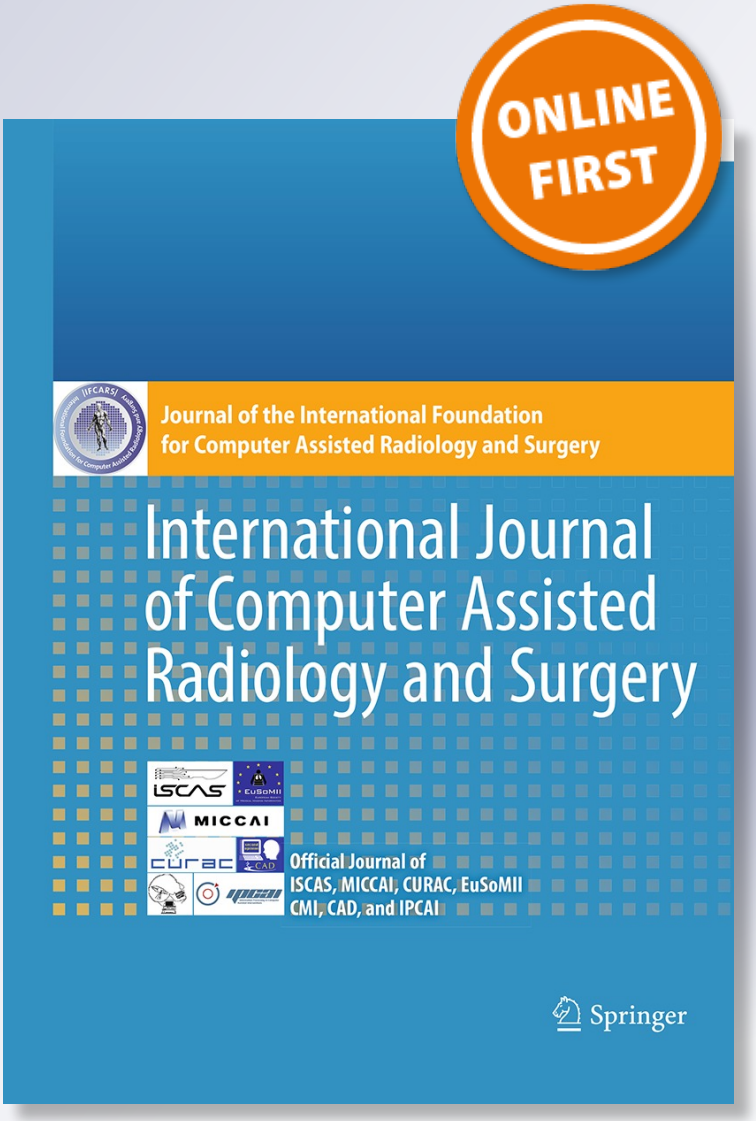
Analytical modeling and feasibility study of a multi-GPU cloud-based server (MGCS) framework for non-voxel-based dose calculations

J. Neylon, Y. Min, P. Kupelian, D. A. Low & A. Santhanam

International Journal of Computer Assisted Radiology and Surgery
A journal for interdisciplinary research, development and applications of image guided diagnosis and therapy

ISSN 1861-6410

Int J CARS
DOI 10.1007/s11548-016-1473-5



Springer

Int J CARS
DOI 10.1007/s11548-016-1473-5

ORIGINAL ARTICLE

Analytical modeling and feasibility study of a multi-GPU cloud-based server (MGCS) framework for non-voxel-based dose calculations

J. Neylon[1] · Y. Min[1] · P. Kupelian[1] · D. A. Low[1] · A. Santhanam[1]

Received: 7 March 2016 / Accepted: 12 August 2016

Abstract
Purpose In this paper, a multi-GPU cloud-based server (MGCS) framework is presented for dose calculations, exploring the feasibility of remote computing power for parallelization and acceleration of computationally and time intensive radiotherapy tasks in moving toward online adaptive therapies.
Methods An analytical model was developed to estimate theoretical MGCS performance acceleration and intelligently determine workload distribution. Numerical studies were performed with a computing setup of 14 GPUs distributed over 4 servers interconnected by a 1 Gigabits per second (Gbps) network. Inter-process communication methods were optimized to facilitate resource distribution and minimize data transfers over the server interconnect.
Results The analytically predicted computation time predicted matched experimentally observations within 1–5 %. MGCS performance approached a theoretical limit of acceleration proportional to the number of GPUs utilized when computational tasks far outweighed memory operations. The MGCS implementation reproduced ground-truth dose computations with negligible differences, by distributing the work among several processes and implemented optimization strategies.
Conclusions The results showed that a cloud-based computation engine was a feasible solution for enabling clinics to make use of fast dose calculations for advanced treatment planning and adaptive radiotherapy. The cloud-based system was able to exceed the performance of a local machine even for optimized calculations, and provided significant acceleration for computationally intensive tasks. Such a framework can provide access to advanced technology and computational methods to many clinics, providing an avenue for standardization across institutions without the requirements of purchasing, maintaining, and continually updating hardware.

Keywords Multi-GPU · GPU · Radiotherapy · Cloud computing

Introduction

Radiotherapy (RT) has been an effective tool in treating many types of cancers, and with recent advancements in dose conformity and improved tumor targeting, significant improvements in treatment efficacy have been observed [1–6]. A major consideration when delivering tumoricidal RT doses is the resultant normal tissue toxicity that results from radiation exposure of healthy anatomy surrounding the tumor target, which may lead to a reduction in bodily functions. Undetected and uncompensated changes in the patient anatomy may lead to a reduction in treatment efficacy and subsequently in the patient's quality of life. Some degree of normal tissue toxicity is currently unavoidable from a planning perspective due to the error margins that are built into the treatment plan to account for the dynamic nature of the patient's anatomy over a course of treatment that may last several weeks. Factors such as weight loss, tumor regression, patient positioning, and posture changes all contribute to changing patient anatomy from day to day [7].

Adaptive radiotherapy (ART) is a critical tool in further improving radiotherapy treatments. ART attempts to enable treatment plans to be altered to compensate for

✉ J. Neylon
jneylon@mednet.ucla.edu

[1] Department of Radiation Oncology, University of California Los Angeles, 200 Medical Plaza, #B265, Los Angeles, CA 90095, USA

Published online: 25 August 2016

changes in patient anatomy due to setup variation, physiological regression, and radiation response [8]. While peers have investigated the potential benefits of ART, clinical implementations of ART are currently limited to off-line studies or require a significant amount of user intervention [6,9]. The computational challenges and increased manpower requirements of ART have inhibited full online capabilities, where the treatment is evaluated daily and plan adaptations computed, validated, and enacted with the patient on the treatment table. In 2007, Xing et al. [10] detailed the difficulty in moving from conventional RT to image-guided RT (IGRT) to off-line ART and finally to online, image-guided ART. They went on to identify three major limiting factors recurring throughout the workflow. These were: (1) reliability: assessing and verifying the accuracy of each process; (2) integration: facilitating the communication of data between processes; and (3) time: the effort and workforce required to complete all processes. In order to feasibly work in a daily clinical workflow, the entire process could not take more than a few minutes to complete or clinical throughput would suffer. Therefore, the algorithms will need to be largely automated and highly accelerated to achieve the necessary performance. There are several possible bottlenecks where acceleration and automation could be applied to minimize the necessary time and labor.

One of the critical components required for broad on-table ART implementation is real-time dose calculation in order to allow the treatment plan to be updated in the time between the daily positioning imaging and beam on without reducing the clinical throughput. Factors that increase computational complexity of re-planning in real-time include (a) complex treatment delivery systems [11,12], (b) treatment plans that aggressively reduce dose to the surrounding organs [8], and (c) complex physiological regression in the patient anatomy [13]. Increasing the computational speedup of the dose convolution will be a critical component to enabling on-table re-planning for ART implementation. The difficulty for a clinic to take full advantage of advancing technology arise from the space required to house the hardware and the necessity to continually upgrade it. Cloud-based computing can give access to far more computational power than would be feasible locally, while constantly expanding and rotating in new hardware as it becomes available.

Recent improvements in near real-time dose convolutions stem from a non-voxel-based (NVB) dose convolution approach and its parallelized implementation [14]. Accelerated dose computations can now approach real-time, but require advanced hardware, such as graphics processing units (GPU). General purpose GPU computing is a rapidly developing field, with new generations released more than once per year. Expanding the calculation to a multi-GPU implementation further improves performance in a linear relationship to the number of GPUs employed [15]. To take full advantage of the potential processing power would require constant hardware updates. Extending these algorithms to run on cloud-based GPU servers would allow clinics to fully utilize these methods without the direct cost or overhead of installing and continuously updating computing hardware. Recent cloud computing works have explored the performance benefits for Monte Carlo-based dose simulations [16–19]. Additionally, Meng et al. [20] explored utilizing Google's MapReduce technology for scaling CT reconstruction using cloud computing.

In 2013, Kagadis et al. [21] presented a thorough introduction to cloud computing, with a review of the recent medical imaging applications. Moore et al. followed this paper in 2014 with a review of advanced computing methods in radiation oncology, and a discussion on promising developments for the near future [22]. These works describe the potential advantages of a cloud computing environment, including access to more extensive computing power and storage, removing the responsibility of maintaining and updating the computer hardware from the clinical institution, and the possibility for shared information between institutions and promoting standardization and collaboration between clinics [23–25]. However, the application of such frameworks utilizing GPUs for radiation therapy purposes remains largely unexplored [26].

In this paper, we present a cloud-based GPU-accelerated dose calculation engine and an analytical model to describe the potential performance of the cloud-based implementation as a function of the distribution of tasks. Specifically, we investigate the feasibility of performing a NVB dose convolution in a multi-GPU cloud-based setup, which provides an additional layer of parallelization for an algorithm already optimized for GPU architecture. In order for this cloud-based framework to be a feasible clinical solution, the performance must be equal or greater than performance on a local machine using a single GPU. Key contributions of the method include (a) optimizing inter-process communication (IPC) between and within cloud server nodes to reduce latencies of memory operations, (b) developing a model to predict performance and acceleration against a single-threaded, single-GPU implementation, and (c) evaluating the feasibility of utilizing a multi-GPU cloud-based server framework for general radiotherapy tasks. In the future, we envision an entire suite of radiotherapy tools instantiated in a multi-GPU cloud-based environment, building off the foundational work presented here, to facilitate the computational performance needed for fully operational online ART.

Materials and methods

Non-voxel-based (NVB) convolution implementation

NVB dose calculation framework has been previously shown as an effective tool for accelerating dose computations, which will be necessary for enabling ART to take into account patient geometry changes and physiological regression. A full description of the NVB dose calculation framework was published in Neylon et al. [14]. However, for clarity, a brief description is presented. The convolution/superposition dose calculation method convolves a Monte Carlo-simulated cumulative–cumulative energy deposition distribution (CCK), with a 3D volume of the total energy released in matter (TERMA) by the primary radiation beam, T.

$$\text{Dose}\,(v) = \int T'\left(v'\right)\text{CCK}\left(\bar{\rho}_{v-v'} * v - v'\right)dv',$$
$$\text{where}\int dv' = \iiint d\varphi d\theta dr, \tag{1}$$

where v was the interaction point, v' was the voxel being sampled, $\bar{\rho}_{v-v'}$ was the heterogeneity correction applied to the kernel, r was the radial component, θ was the zenith angle, and φ was the azimuthal angle. What makes this convolution computationally challenging is that the sampling of the CCK is dependent on radiological effective depth. For each convolution direction, defined by the polar and azimuthal angles, calculating the contributions from the convolution requires walking along the convolution ray, sampling the density, and accumulating the effective radiological depth. Because of this, the convolution/superposition algorithm cannot be parallelized as efficiently as standard image processing convolutions.

The NVB algorithm was developed to optimize the convolution/superposition algorithm for GPU architecture, focusing on maximizing the efficiency of GPU memory usage and access patterns. The algorithm looped over the convolution rays and converted the data volumes to a NVB coordinate system by ray tracing through the volumes along each convolution direction. The data required for the convolution (density, TERMA, and the CCK) can be precomputed in a separate process.

Extending the NVB algorithm to multi-GPU

Enabling the NVB dose calculation framework with the versatility and scalability of a cloud computing framework could give any clinic the ability to instantiate ART in their therapeutic workflow. The NVB dose convolution algorithm lent itself well to a multi-GPU implementation. Each convolution direction was already being calculated independently and summed afterward to obtain the final dose distribution. Thus, the GPU convolution for a single direction could be extracted as a separate process and distributed among the available GPU devices. The results from all convolution directions can be consolidated using IPC and then summed and normalized to produce the final dose distribution.

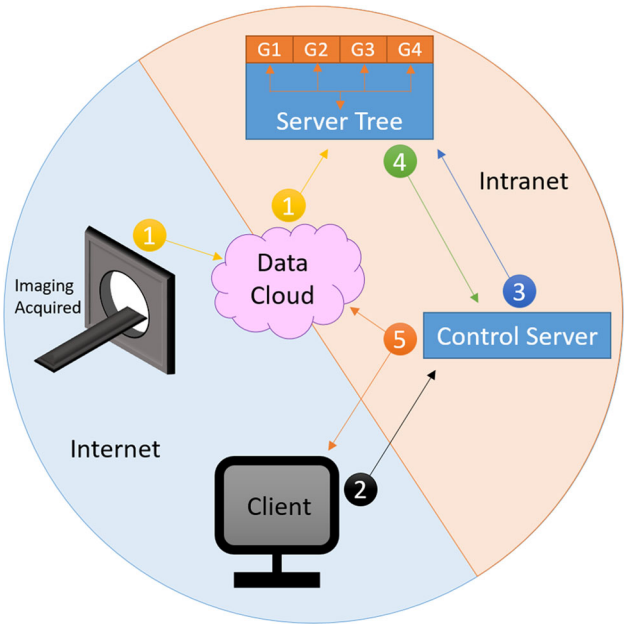


Fig. 1 A schematic representation of a potential workflow utilizing a MGCS framework. After acquisition (*1*), imaging data syncs to a cloud database. Once synced, the data are available for treatment planning and the planner may signal a control server (acting as the client) to initiate computation (*2*). Control distributes the workload among available resources (*3*). After computation, results are accumulated and normalized by control (*4*), and final results are returned to client and pushed to the cloud database so the planner can review the dose distribution (*5*)

Figure 1 provides a schematic representation of a possible workflow using the MGCS framework for dose computations. After the patient's simulation CT is acquired, the imaging data would be synced with a cloud storage database with direct high-speed access to all server machines (step 1 in Fig. 1). After the data have synced, and treatment planning has commenced, the planner (as remote client) would signal the MGCS framework that they wished to perform calculations, and a single server node would be assigned as the control server (step 2 in Fig. 1). Calculation parameters are communicated between the remote client and the control server. The control server would then intelligently distribute tasks to other computational servers in the MGCS network (step 3 in Fig. 1). For this purpose, a binary branching inter-server communication setup is employed, as shown in Fig. 2. The control server first communicates the dose computation parameters with the selected server nodes (in Fig. 2 example setup, server nodes 1 and 2). These servers then communicate with other servers in a similar manner.

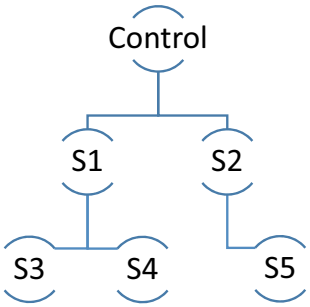


Fig. 2 A schematic of the branching server framework. This method allows parallel accumulation of results between servers, reducing the number of data transfers from a linear relation with the number of servers, to a log relation

For the example setup, server node 1 communicates to server nodes 3 and 4 with the required dose computation parameters, and server node 2 then further propagates to server node 5. Once the computation is completed asynchronously by all the server nodes, data transfers of the 3D dose distribution occur in parallel back along their communication pipeline to the control server (step 4 in Fig. 1). Finally (step 5 in Fig. 1), the control server performs final 3D dose summation and normalization of results before sending them back to the client in the form of (a) DVH curves representing the dose to be delivered and (b) specific dosimetric endpoints for critical structures. Additionally, the dose distribution would also be synced to the cloud-based database for 3D image viewing and interaction. This workflow utilizes multi-level parallelism and would be adaptively scalable to the resources available.

Theoretical model for the multi-GPU cloud server acceleration

In order to quantify the feasibility of performing a cloud-based multi-GPU implementation, an analytical model of the computational acceleration is required. Such a model will also inform the amount of scalability required for specific ART treatment plans. A generalized methodology was developed to estimate performance gains for MGCS implementation with Eqs. 2–17. The total computation, presented in Eq. 13, can be divided into seven stages: (0) signal from client to control, (1) server nodes load precomputed data, (2) initialize GPU memory, (3) perform computations, (4) aggregate results, (5) consolidate results from server nodes to control, and (6) send final results back to client. Table 1 summarizes the variables in the below equations.

Let t_0 represent the time taken for the client signal to control and the subsequent chain of inter-server communication. Since the initial signaling between the client and the control does not include large data transfers, the value for t_0 was a few milliseconds. Let t_1 represent the time taken by the individual server nodes to load the 3D image dataset from cloud storage. It is defined as

$$t_1 = \frac{f_L d}{v_L}, \tag{2}$$

where d represents the total data size, f_L represents the ratio of the precomputed data size and the result data size, and v_L represents the disk read rate. Let t_2 represent the total time taken for initializing the device memory in each GPU (a_A), defined as

$$t_2 = a_A, \tag{3}$$

The computation time t_3 is defined as a function of the time required for computing a single-field dose distribution, T, multiplied by the total number of fields in the complete dose plan, n_F, divided by the product of the number of server nodes, N_S, and the number of GPUs per server node, N_G. It is defined as

$$t_3 = \left(\frac{n_F T}{N_S N_G}\right), \tag{4}$$

Let t_4 represent the time taken for the accumulation of the dose distribution results from the individual GPUs of each server process. It is defined as a product of the summation of peer-to-peer data retrieval time (a_R) and the GPU-based dose

Table 1 Definition of variables

Number of server processes	N_S	Ratio of precomputed data size to results size	f_L
Number of devices per server process	N_G	Factor of acceleration from multiple streams	f_M
Result data size	d	Device memory allocation time	a_A
Compressed result data size	d_s	Device peer-to-peer setup time	a_S
Server read from disk speed	v_L	Device peer-to-peer retrieval time	a_R
Local server network speed	v_N	Data summation kernel time	k_S
Cloud/client internet connection speed	v_I	Data normalization kernel time	k_N
Original single-field computation time	T	Number of fields in dose calculation	n_F

summation time (k_S) and the number of additional GPUs utilized by the server process.

$$t_4 = (a_R + k_S)\,(N_G - 1) \tag{5}$$

Let t_5 represent the time taken for retrieving the results from all server nodes and normalizing the final result. It is defined as

$$t_5 = \frac{d\,N_s}{v_N} + k_S\,(N_S - 1) + k_N \tag{6}$$

where v_N represents the inter-server network speed, k_S and k_N are the kernel execution times for summing and normalizing the results, respectively.

Finally, let t_6 represent the time taken to send the final computed results from the control server to the remote client, where v_I is the Internet connection speed between the MGCS control server and the remote client.

$$t_6 = \frac{d}{v_I} \tag{7}$$

Algorithm optimizations

Minimizing networks data transfers—inter-server communication

Unix sockets [27] were employed for inter-server communication. Establishing the socket connection was a simple procedure requiring only an auto-generated port number and the Internet Protocol address of the server node. The maximum data rate transfer on any network was fixed, and so, the only way to reduce the data transfer time was to minimize the number of data transfers, and the size of each data transfer. In order to minimize the number of data transfers, each server process was configured to run multiple convolutions and sum the results, which limited the memory transfers to one per server node. The workload distribution by the control was optimized to scale to any number of server nodes and equally distribute the convolution directions. To minimize the size of each data transfer, only the sub-volume involved in the calculation was transferred as opposed to the full 3D data matrix. For a typical CT scan, over half of the image data are empty area outside of the patient's body, and only a small portion of the patient anatomy is actually involved in the dose computation. The anatomy involved in the calculation was defined by the precomputed TERMA (total energy released in matter) matrix. The equations for terms t_2, t_4, t_5, and t_6 were modified to incorporate the compressed data size, d_c. Further reduction was achieved by converting the dose results from floating point (4 bytes) to short integer type (2 bytes). Extensive systematic studies were performed to test the accuracy of this method, and the results presented in §III.A confirm the expected precision of five significant figures. Terms t_5 and t_6 were reduced by a factor of two to reflect the change in data type.

Lastly, while increasing the number of server processes reduced the computation time, it also increased the time required to copy results back to the control. The bandwidth for control-node network data transfers saturated, which resulted in serialization of data retrieval. Therefore, instead of copying the data from each server node sequentially to the control, the branching method described from Fig. 2 was introduced. For three or more server nodes, a copy and summation may occur between two nodes concurrently with the control's data retrieval from a third node. This effectively reduced the number of copies to the control from N_s to $log(N_s)$. The logarithmic relationship was applied into term t_5.

Incorporating the above strategies into the terms t_2, t_4, t_5, and t_6 resulted in the following modifications:

$$t_2 = a_A \left(\frac{d_c}{d}\right), \tag{8}$$

$$t_4 = (a_R + k_S)\,(N_G - 1)\left(\frac{d_c}{d}\right) \tag{9}$$

$$t_5 = \frac{d_c \log N_S}{2v_N} + (k_S\,(N_S - 1) + k_N)\left(\frac{d_c}{d}\right) \tag{10}$$

$$t_6 = \frac{d_c}{2v_I} \tag{11}$$

Optimizing IPC methods—intra-server communication

The bandwidth for transferring data between the server processes was limited by the network interconnect speed. Copying memory using peer-to-peer access between GPUs has a much higher bandwidth than transferring memory between CPU processes. Accessing multiple GPUs through a single server process also requires less overhead than launching a server process for each device. Using this method, the convolution results were accessed peer-to-peer using CUDA IPC memory handles to consolidate the results from each GPU before copying back to the CPU in the parent server process and consolidating the results through inter-server communication. This provided the least amount of memory transfer on the CPU side, while maximizing the parallelism on the GPU side. The term t_2 was modified to include the CUDA IPC setup time, a_S, for each additional GPU as follows:

$$t_2 = (a_A + a_S\,(N_G - 1))\left(\frac{d_c}{d}\right), \tag{12}$$

Optimizing memory concurrency—inter-GPU communication

Methods explored for optimizing memory concurrency included forked processes, multiple streams per GPU, multiple GPUs per process, and multiple machines. Only a single CUDA context may run on a GPU device at a time; launching multiple processes on a single device requires context switching. Context switching resulted in sequential kernel calls and additional overhead for scheduling tasks. Concurrent execution on the GPU was only possible using multiple streams within a single process. Operations in separate streams can overlap, while operations within a single stream will execute in order. Concurrent execution on the GPU using streams is limited only by the device resources, such as memory size and number of processing cores. However, some consideration must be given to avoid blocking calls, such as copying memory between the CPU and GPU, which will synchronize streams and diminish parallel performance. We thus modify the term t_3 as follows:

$$t_3 = f_M\left(\frac{n_F T}{N_S N_G}\right), \tag{13}$$

where f_M represents the factor of acceleration from the multiple stream usage.

Concurrency was further tested by forking the control process to communicate with each server node simultaneously, instead of looping over the server processes and signaling them sequentially. Similarly, the server process was forked for each GPU under its control, parallelizing the CPU-GPU communication and transferring data between devices using CUDA IPC memory handles. For this method, an array was allocated on GPU0 for each of the forked processes. Memory handles were created and copied into a mapped memory array accessible by all processes. After computation, the results were copied through peer-to-peer access.

Optimized model for total MGCS computation time

Applying the optimized representations for each term, the total time (t_{MGC}) of the MGCS implementation of an algorithm was defined as:

$$\begin{aligned} t_{\mathrm{MGC}} = {} & \frac{f_L d}{v_L} + \frac{f_M n_F T}{N_S N_G} + \left(\frac{d_c}{d}\right)[a_A \\ & + (a_S + a_R + k_S)(N_G - 1) + k_S(N_S - 1) + k_N] \\ & + \frac{d_c \log N_S}{2v_N} + \frac{d_c}{2v_I} \end{aligned} \tag{14}$$

The theoretical acceleration (A_{MGC}) using the MGCS framework was the ratio of the original algorithm's execution time and Eq. 14, giving the following final form.

$$A_{\mathrm{MGC}} = \frac{n_F T}{t_{\mathrm{MGC}}} \tag{15}$$

When expanding to a multi-GPU implementation, a direct linear relationship between the acceleration and the total number of GPU devices being employed was expected. For the MGCS framework, the overhead of data transfers and other memory operations must also be considered. Therefore, the limit of acceleration should approach a direct linear relationship for computationally heavy tasks where the calculation time on a single GPU was much larger than the overhead of memory operations in the MGCS framework. This can be shown by rewriting Eq. 14 under the assumption that term 3 is much larger than the sum of all other terms, C:

$$t_{\mathrm{MGC}} = \frac{n_F T}{N_S N_G} + C \approx \frac{n_F T}{N_S N_G}; \text{ when} \frac{n_F T}{N_S N_G} \gg C \tag{16}$$

Substituting into Eq. 16 gives the theoretical limit for acceleration of the MGCS framework:

$$A_{\mathrm{MGC}} = \frac{n_F T}{t_{\mathrm{MGC}}} \approx n_F T\left(\frac{N_S N_G}{n_F T}\right) = N_S N_G \tag{17}$$

Results

For numerical studies, the client process was run locally on 64-bit Linux with an 8-processor Intel Core i7 3.6GHz CPU. A set of three network server nodes were also running 64-bit Linux with four NVIDIA GeForce 980 GPUs running CUDA 6.5. An additional server node consisted of two NVIDIA GeForce 780 Ti GPUs, running CUDA 6.5. The server nodes communicated through a 1 Gbps Ethernet backbone.

Minimizing networks data transfers—inter-server communication

To reduce the data size and minimize data transfer time, the data were cropped around the active volume. The size of the cropped volume was determined by the precalculated TERMA volume plus an additional penumbra region to account for scatter. TERMA calculations generally completed in less than a millisecond. The original volume was a 256-mm-sided cube, comprised of 1-mm isotropic voxels. The ratio of the reduced data size to the original data size followed a near linear fit as a function of the cross-sectional area of the irradiation field. For example, for a 100×100-mm field, the data size could be reduced from 67 MB to 20 MB. The active volume included the main field as well as any voxels in the penumbra that would receive dose due to scattering. Converting from float to short data type further reduced the data size by a factor of 2, resulting in a reduction

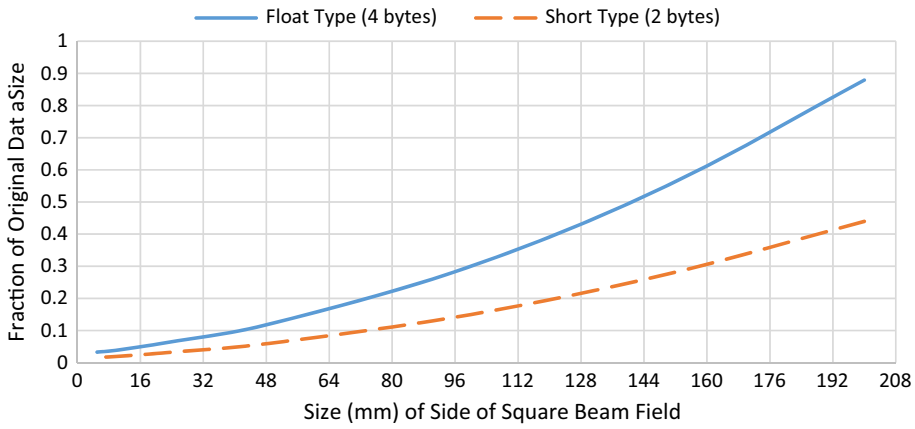


Fig. 3 Potential reduction in data transfer size as a function of field size. Results show the fraction of the total volume for a 256-mm cubic dataset of 1-mm isotropic voxels

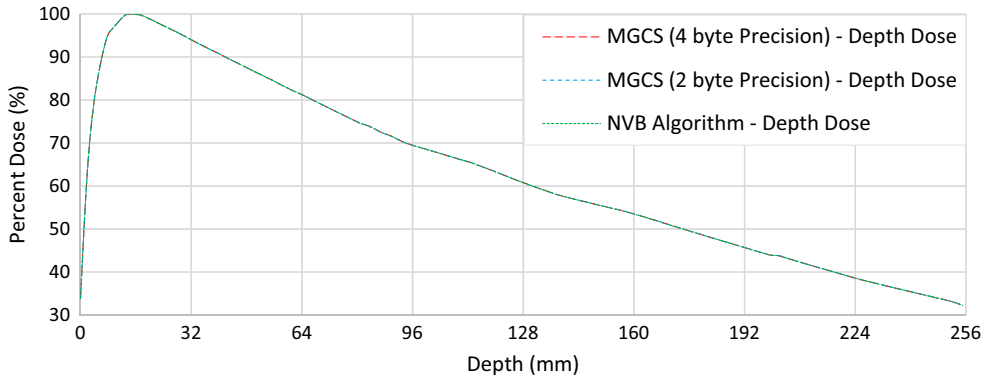


Fig. 4 Accuracy of multi-GPU cloud server implementation. Depth dose curves for the original NVB algorithm, the 4-byte float precision MGCS implementation, and the 2-byte short precision MGCS implementation. Plots calculated using a 100×100 mm square field on a standard mediastinum phantom with 1-mm isotropic voxels and a size of 256 mm along each axis. The MGCS configuration depicted utilized 2 server nodes, with each node utilizing 2 GPUs

of 85 % for a 100×100 mm field. Figure 3 shows the potential reduction in data size for both float-type and short-type data.

Figure 4 shows a comparison of the depth dose profile through isocenter of a standard mediastinum phantom for the original NVB algorithm, the MGCS implementation using both floating point and short integer precision. There was minimal error when reducing precision, with a maximum of 4e-3 %. The error between the NVB algorithm and the MGCS implementation along the depth dose curve was less than 1e-4 %.

Numerical analysis of the MGCS acceleration

The observed results were also compared to the predicted results from the equations in section II.D. Table 2 shows the approximate values for GPU-specific (GTX 780 Ti) run time and network variables. The algorithm, specifically the convolution kernel, required too many GPU resources for concurrent execution, resulting in no performance gains from employing multiple streams, setting the acceleration factor, f_M, to 1.

Table 2 Estimate values of variables for the NVB implementation on the multi-GPU cloud server framework

f_M	1	a_A	35 ms
T	400 ms	a_S	10 ms
n_F	1	a_R	5 ms
v_N	0.125 MB/ms	k_S	5 ms
v_I	0.125 MB/ms	k_N	5 ms

Table 3 shows the strong agreement between the predicted and observed results when increasing the number of GPUs on a single server process, with estimations within 5 % of the observed performance. Using this simple calculation, many different combinations of resources can be compared to find the optimal distribution of the workload.

Theoretically, increasing the number of server processes would reduce the total computation time by a linear proportion and this was seen in the convolution times ($t3$) of the

Table 3 Comparison between predicted and observed performance for a single server while increasing the number of GPUs

Number of GPUs	Predicted performance (ms)	Observed performance (ms)	% Difference
1	485	479	1.24
2	330	335	1.52
3	288	287	0.35
4	274	260	5.11

model. Figure 5a shows the model-predicted response for GPU initialization ($t2$), convolution computation ($t3$), and accumulation and normalization of the dose results ($t4$–$t6$) for a single-field convolution calculation as a function of the number of servers in the MGCS framework, with each server employing two GPUs. Figure 5b shows the actual timing data acquired from experiments. Similarly, Fig. 6 compares the predicted and observed response to an increasing number of GPUs being employed by a single server. The graphs show good correspondence between the predicted and observed values in both cases.

Figures 5 and 6 also illustrate how memory operations can dominate for small computations, such as a single-field dose convolution. Overall acceleration was inhibited by approximately equal contributions of the convolution calculations and memory operations to the total MGCS computation. Despite the additional latencies, the MGCS framework was still able to accelerate the single-field dose convolution by 2x. This illustrates that the MGCS framework, with the optimization strategies described above, was able to outperform a local single-GPU system for even the most efficient, well-optimized processes. However, to fully utilize the increased computational power of the MGCS framework, the algorithm's computation time should be dominated by the calculation stage.

Optimizing performance

For volumetric arc therapy (VMAT) or tomotherapy planning, which regularly utilize 180–225 fields, the time required to calculate a complete dose distribution is much greater than the memory operations. Here, the MGCS framework approaches the linear acceleration limit, defined by the total number of GPUs. Figure 7 displays MGCS acceleration response to an increasing number of fields, when employing 1, 2, 3, or 4 servers, each utilizing 4 GPUs. For a 250-field dose plan, the MGCS framework reached peak accelerations of 3.98x, 7.88x, 11.7x, and 15.4x, for their respective number of servers. The biggest factors contributing to the maximum achievable acceleration were the original algorithm's computation time, and its proportion to the data transfer time as dictated by the size of the results. More computationally expensive algorithms such as many-field dose calculations have the largest potential benefit from the MGCS framework, while minimizing the data transfers between the server nodes increased the framework efficiency.

Figure 8 shows the potential performance gains from utilizing an MGCS implementation of the NVB dose convolution algorithm for computationally intensive, many-field calculations. The single-GPU implementation was already providing a significant performance improvement over the

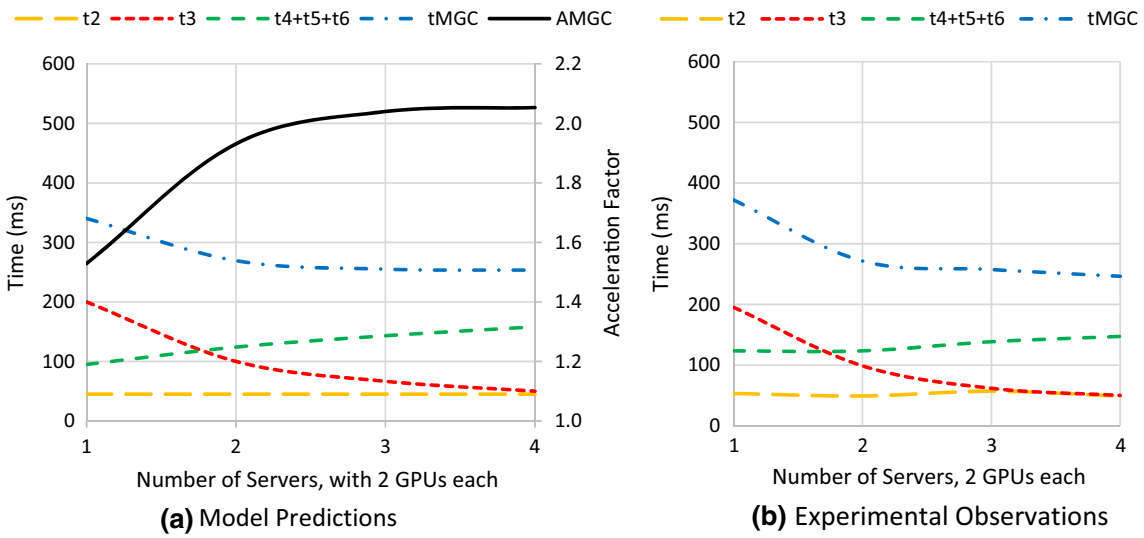


Fig. 5 **a** Relationship between performance and increasing number of servers in the MGCS configuration for individual terms as predicted by the model. **b** Observed performance of individual terms when increasing the number of servers in the MGCS framework. Values reported for a single-field dose calculation with parameters described in Table 2

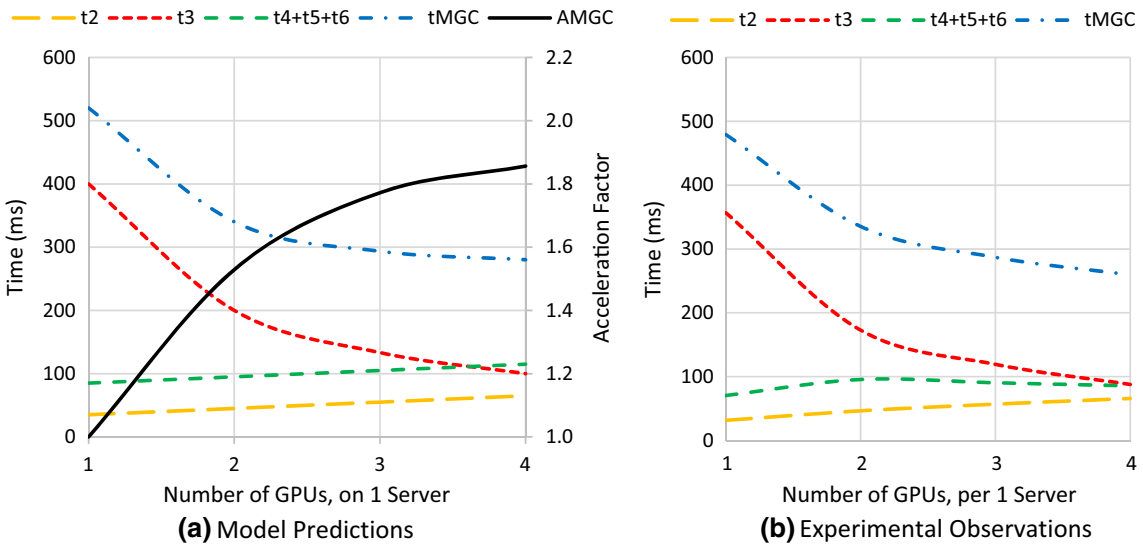


Fig. 6 **a** Relationship between performance and increasing number of GPUs on a single server in the MGCS configuration for individual terms as predicted by the model. **b** Observed performance of individual terms when increasing the number of GPUs on a single server in the MGCS framework. Values reported for a single field dose calculation with parameters described in Table. 2

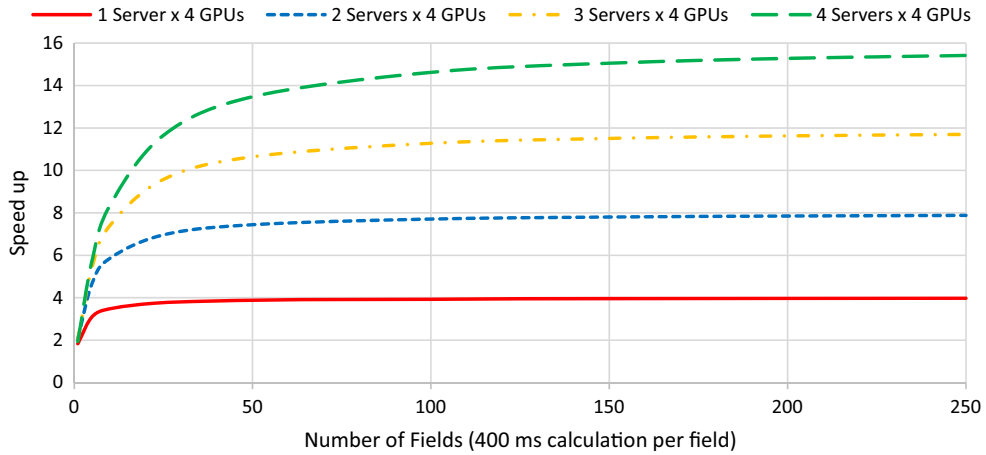


Fig. 7 Acceleration as a function of the number of fields (each a 400-ms calculation on a single GPU) for different configurations of resources. Predicted maximum acceleration equal to the total number of GPUs

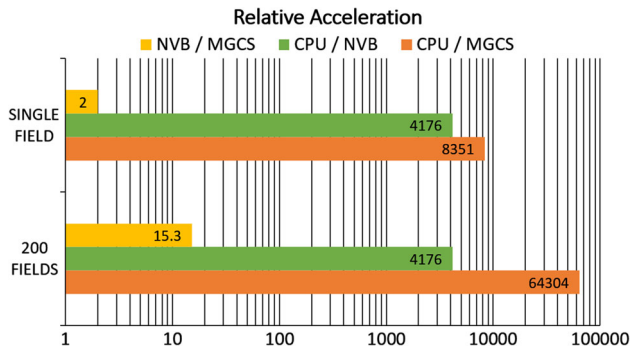


Fig. 8 Performance comparisons between a CPU dose convolution, a single-GPU implementation of the NVB dose convolution, and an MGCS implementation of the NVB dose convolution for a single field and for many fields

CPU implementation, offering over 4000x speed up for a single-field convolution. The single-GPU NVB algorithm reduced the computation for a single field from several minutes to a fraction of a second. The MGCS implementation was able to improve upon an already optimized algorithm, and increase performance by another factor of 2. For a many-field calculation, the single-GPU implementation of the NVB algorithm again provided over 4000x speed up compared to a CPU implementation. With the additional 15x speed of attained from the multi-GPU distribution, the total performance gains of the MGCS implementation would reach over 64,000x speedup compared to a single-threaded CPU convolution. This should shorten the calculation time to a matter of seconds as opposed to hours.

Discussion

Feasibility of MGCS implementation

The feasibility of a MGCS framework for remote dose calculations was investigated in this paper. The proposed method could potentially enable clinics to make use of continually advancing technologies without the requirements of purchasing, maintaining, and frequently upgrading hardware. With the price/performance ratio getting smaller every day, more and more GPUs can be integrated with the cloud computing framework to progressively improve the computation speed. Additionally, a cloud-based solution holds the potential for much greater computing power than could feasibly be installed in the limited space of a clinic, and provide services to multiple clinics with a possibility for increased standardization across institutions.

An analytical model to estimate the potential acceleration available using the MGCS framework was also presented. The model took into account the workload distribution in the MGCS framework, the original algorithm parameters, and the inter-GPU and inter-server communication latencies. The estimated computing time was numerically validated using a 4 server/14 GPU cloud computing setup. The model-predicted performance and experimentally-observed performance matched within 5 %. In addition, discrepancies between the dose distributions calculated by a local, single-GPU implementation and the MGCS implementation were observed to be negligible.

Without the optimization strategies (§II.D.1–§II.D.3), the additional overhead of memory operations and IPC of an MGCS framework would result in reduced performance. From our analytical model, we estimated that a brute force method of cloud-based implementation would result in 2x slower computation times or worse. In addition, an increase in the number of GPUs will only result in added complexity and will not improve the performance. In contrast, the MGCS implementation presented in the manuscript achieved >2x faster computation times even for highly optimized computations, such as a single-field convolution which takes less than 350 ms locally on a single GPU. This results showed the MGCS implementation to be >4x better than the estimated brute force method in this case.

For computationally heavy tasks, such as the many-field dose plans of VMAT and tomotherapy, the MGCS framework approaches the theoretical acceleration limit directly proportional to the total number of GPUs being utilized. For instance, for a 4 server configuration with 4 GPUs each (16 total GPUs), the modeled acceleration of the MGCS framework peaked at 15.4x speedup.

MGCS performance falls just short of the theoretical limit of 16x due to data transfer and other memory operation overhead. In this manuscript, these aspects were minimized by reducing the number and size of the data transfers. Another incremental improvement could come from increasing the speed of inter-server communications. By replacing the 1 Gbps ethernet network with a 16 Gbps or PCIe3 interconnect backbone, data transfer times between the server nodes can be reduced. Infiniband offers another alternative for inter-server communication, replacing the socket data transfers with direct memory copies at an increased bandwidth. However, one of the key conclusions presented from our theoretical model was that the potential benefit of the MGCS implementation was maximized when the compute effort far outweighed the memory operations, making the interconnect speed less of a priority.

Clinical considerations and future work

Internet speed can be unreliable due to bandwidth limitations or slower connection speeds on the client end. In this case, DVH data and specific dose endpoints for critical targets and organs at risk could be sent directly back to the client, while the full dose distribution was synced with cloud storage, as described in Fig. 1. This would avoid any significant delays from transferring large amounts of data.

To maintain scalable performance for increasingly computationally heavy tasks, future work will focus on more efficiently distributing the workload among the server nodes. Distributing the workload of the incoming client task will require the control server to analyze the optimal number of required server processes and GPUs to utilize, using the model to estimate the best possible acceleration. The control will then query the server tree to identify available resources and idle server nodes. Additionally, data security and encryption will also be included to better quantify the feasibility of a cloud-based dose computing framework, and their subsequent effect on performance must be analyzed.

Encryption of the communication between the server nodes as well as the client server nodes is an essential step. While our current implementation has not dealt with the data encryption steps, the roll of encryption has been well documented with respect to cloud computing in the medical field, with solutions continually improving in both security and performance [21,28,28,30–32], but the impact on performance for specific applications will need to be explored. Future work would focus on developing advanced encryption mechanism that will be aware of the cloud computing nature and will facilitate a fast encryption. We envision that the proposed work also motivates encryption experts to design such algorithms as it may have a direct impact in improving treatment quality.

Building upon the MGCS dose computation

The feasibility of utilizing an MGCS framework for dose convolutions was investigated in this manuscript, and there are several computationally expensive tasks in radiotherapy that could benefit from access to the power and throughput of an MGCS framework. Of particular importance are inverse optimization strategies. Many treatment methodologies utilize inverse optimization to determine radiation delivery. For conventional treatment planning, the speed of such techniques is not a concern and can take several hours without interrupting the clinical workflow. However, for online adaptive re-planning, inverse optimization would need to be completed in a matter of minutes. The iterative nature of inverse optimization demands many repeated calculations, which quickly becomes time consuming. Distributing the work over several server nodes and utilizing GPU parallelization would greatly accelerate the optimization process. As discussed in the previous section, a brute force method of pushing these calculations in the cloud can result in longer computation times due to the additional overhead of memory transfer operations and inter-process communication, whereas the optimization strategies employed in the MGCS implementation make a cloud-based solution feasible and provide the desired performance boost and scalability.

More complex planning methods further emphasize the need for access to extensive computational networks to accelerate these tasks to a point where it becomes feasible to perform on-table adaptive re-planning. Intensity-modulated therapies require optimization of the motion of the multileaf collimators. Small physiological changes can completely alter the MLC sequence to better focus on the target anatomy. Another example is 4π treatment planning, which optimizes beam delivery in three dimensions, as opposed to the traditional axial in-plane optimizations [11,12]. In each of these advanced treatment planning methods, the optimization process can be directly wrapped around the MGCS dose convolution, utilizing the speed and power in scenarios where many fields must be calculated and optimized iteratively.

The impact of a MGCS framework is applicable beyond dose computations as well. Several areas have been identified by peers as potential bottlenecks that would impede online ART, including deformable image registration, contour segmentation, and dose propagation [10]. We envision a fully realized MGCS framework coupled with meticulously parallelized algorithms for image registration, and dose propagation will be able to perform daily adaptive adjustments within the time frame of a few minutes after the patient's daily imaging is acquired.

Conclusion

The multi-GPU cloud-based dose convolution presented in this paper could improve radiotherapy treatment outcomes by facilitating more frequent re-planning due to its potential for accelerating calculations. The MGCS framework described in this work shows that a multi-GPU cloud-based solution is feasible for accelerating radiotherapy tasks from a computational perspective, and provides a foundational methodology for future clinical tools.

Acknowledgments This work was funded in part by Varian and The National Science Foundation.

Funding The funding for this study consisted of contributions by Varian, the National Science Foundation (1200579), and the Ken & Wendy Ruby Foundation award.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Human and animal rights statement This article does not contain any studies with human participants or animals performed by any of the authors.

Patient data This article does not contain patient data.

References

1. Foroudi F, Wong J, Kron T, Rolfo A, Haworth A, Roxby P, Thomas J, Herschtal A, Pham D, Williams S, Tai KH, Duchesne G (2011) Online adaptive radiotherapy for muscle-invasive bladder cancer: results of a pilot study. Int J Radiat Oncol 81(3):765–771
2. Lindegaard J, Fokdal L, Nielsen S, Juul-Christensen J, Tanderup K (2013) Mri-guided adaptive radiotherapy in locally advanced cervical cancer from a nordic perspective. Acta Oncol 52(7):1510–1519
3. Nijkamp J, Marijnen C, Herk MV, Triest BV, Sonke J (2012) Adaptive radiotherapy for long course neo-adjuvant treatment of rectal cancer. Radiother Oncol 103(3):353–359
4. Schwartz D, Garden A, Thomas J, Chen Y, Zhang Y, Lewin J, Chambers M, Dong L (2012) Adaptive radiotherapy for head-and-neck cancer: initial clinical outcomes from a prospective trial. Int J Radiat Oncol 83(3):986–993
5. Tuomikoski L, Collan J, Keyrilainen J, Visapaa H, Saarilahti K, Tenhunen M (2011) Adaptive radiotherapy in muscle invasive urinary bladder cancer—an effective method to reduce the irradiated bowel volume. Radiother Oncol 99(1):61–66
6. Capelle L, Mackenzie M, Field C, Parliament M, Ghosh S, Scrimger R (2012) Adaptive radiotherapy using helical tomotherapy for head-and-neck cancer in definitive and postoperative settings: initial results. Clin Oncol 24(3):208–215
7. Castadot P, Lee JA, Geets X, Gregoire V (2010) Adaptive radiotherapy of head and neck cancer. Semin Radiat Oncol 20(2):84–93. doi:10.1016/j.semradonc.2009.11.002
8. Qi XS, Santhanam A, Neylon J, Min Y, Armstrong T, Sheng K, Staton R, Pukala J, Pham A, Low D, Lee S, Steinberg M, Manon R, Chen A, Kupelian P (2015) Near real-time assessment of anatomic and dosimetric variations for head-and-neck radiation therapy via

graphics processing unit-based dose deformation framework. Int J Radiat Oncol 92(1):415–422
9. Veiga C, McClelland J, Moinuddin S, Lourenco A, Ricketts K, Annkah J, Modat M, Ourselin S, D'Souza D, Royle G (2014) Toward adaptive radiotherapy for head and neck patients: feasibility study on using CT-to-CBCT deformable registration for "dose of the day" calculations. Med Phys 41(3):031703. doi:10.1118/1.4864240
10. Xing L, Siebers J, Keall P (2007) Computational challenges for image-guided radiation therapy: framework and current research. Semin Radiat Oncol 17(4):245–257
11. Dong P, Lee P, Ruan D, Long T, Romeijn E, Low D, Kupelian P, Abraham J, Yang Y, Sheng K (2013) 4PI noncoplanar stereotactic body radiation therapy for centrally located or larger lung tumors. Int J Radiat Oncol 86(3):407–413
12. Dong P, Lee P, Ruan D, Long T, Romeijn E, Yang Y, Low D, Kupelian P, Sheng K (2013) 4PI non-coplanar liver sbrt: a novel delivery technique. Int J Radiat Oncol 85(5):1360–1366
13. Neylon J, Qi X, Sheng K, Staton R, Pukala J, Manon R, Low DA, Kupelian P, Santhanam A (2015) A GPU based high-resolution multilevel biomechanical head and neck model for validating deformable image registration. Med Phys 42(1):232–243. doi:10.1118/1.4903504
14. Neylon J, Sheng K, Yu V, Low D, Kupelian P, Santhanam A (2014) A nonvoxel-based dose convolution/superposition algorithm optimized for scalable gpu architectures. Med Phys 41(10):101711
15. Santhanam AP, Y M, N H, P N, M SL, Kupelian P (2011) A multi-GPU real-time dose simulation software framework for lung radiotherapy. Int J Comput Assist Radiol Surg 7(5):705–719
16. Constantin M, Sawkey D, Mansfield S, Svatos M (2011) SU-E-E-05: the compute cloud, a massive computing resource for patient-independent monte carlo dose calculations and other medical physics applications. Med Phys 38(6):3392
17. Miras H, Jiminez R, Miras C, Goma C (2013) CloudMC: a cloud computing application for Monte Carlo simulation. Phys Med Biol 58:N125–N133
18. Poole C, Cornelius I, Trapp J, Langton C (2012) Radiotherapy Monte Carlo simulation using cloud computing technology. Australas Phys Eng 35(4):497–502
19. Poole C, Cornelius I, Trapp J, Langton C (2011) Technical Note: Radiotherapy dose calculations using GEANT4 and the Amazon Elastic Compute Cloud. arXiv:1105.1408
20. Meng B, Pratx G, Xing L (2011) Ultrafast and scalable cone-beam CT reconstruction using MapReduce in a cloud computing environment. Med Phys 38(12):6603–6609. doi:10.1118/1.3660200
21. Kagadis GC, Kloukinas C, Moore K, Philbin J, Papadimitroulas P, Alexakos C, Nagy PG, Visvikis D, Hendee WR (2013) Cloud computing in medical imaging. Med Phys 40(7):070901. doi:10.1118/1.4811272
22. Moore KL, Kagadis GC, McNutt TR, Moiseenko V, Mutic S (2014) Vision 20/20: automation and advanced computing in clinical radiation oncology. Med Phys 41(1):010901. doi:10.1118/1.4842515
23. D'Haese PF, Konrad PE, Pallavaram S, Li R, Prassad P, Rodriguez W, Dawant BM (2015) CranialCloud: a cloud-based architecture to support trans-institutional collaborative efforts in neurodegenerative disorders. Int J Comput Assist Radiol Surg 10(6):815–823. doi:10.1007/s11548-015-1189-y
24. Silva LA, Costa C, Oliveira JL (2013) DICOM relay over the cloud. Int J Comput Assist Radiol Surg 8(3):323–333. doi:10.1007/s11548-012-0785-3
25. Silva LA, Costa C, Oliveira JL (2012) A PACS archive architecture supported on cloud services. Int J Comput Assist Radiol Surg 7(3):349–358. doi:10.1007/s11548-011-0625-x
26. Griebel L, Prokosch H, Kopcke F, Toddenroth D, Christoph J, Leb I, Engel I, Sedlmayr M (2015) A scoping review of cloud computing in healthcare. Bmc Med Inform Decis 15(17):1
27. Stevens WR (1998) UNIX network programming, vol 1–2, 2nd edn. Prentice Hall PTR, Upper Saddle River
28. Kanso A, Ghebleh M (2015) An efficient and robust image encryption scheme for medical applications. Commun Nonlinear Sci 24(1–3):98–116. doi:10.1016/j.cnsns.2014.12.005
29. Li M, Yu S, Zheng Y, Ren K, Lou W (2012) Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE Trans Parallel Distrib Syst 24(1):131–143. doi:10.1109/TPDS.2012.97
30. Shao Z, Yang B, Zhang W, Zhao Y, Wu Z, Miao M (2015) Secure medical information sharing in cloud computing. Technol Health Care 23(Suppl 1):S133–S137. doi:10.3233/thc-150945
31. Page A, Kocabas O, Soyata T, Aktas M, Couderc JP (2015) Cloud-based privacy-preserving remote ECG monitoring and surveillance. Ann Noninvasive Electrocardiol. 20(4):328–337. doi:10.1111/anec.12204
32. Barhamgi M, Bandara AK, Yu YJ, Belhajjame K, Nuseibeh B (2016) Protecting privacy in the cloud: current practices. Future Dir Computer 49(2):68–72