Toward real-time Monte Carlo simulation using a commercial cloud computing infrastructure

Phys. Med. Biol. **56** (2011) N175–N181 doi:10.1088/0031-9155/56/17/N02

NOTE

Toward real-time Monte Carlo simulation using a commercial cloud computing infrastructure[*]

Henry Wang[1,3], Yunzhi Ma[2], Guillem Pratx[2] and Lei Xing[2]

[1] Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA
[2] Department of Radiation Oncology, Stanford University School of Medicine, Stanford, CA 94305-5847, USA

E-mail: hwang41@stanford.edu

Received 15 April 2011, in final form 20 July 2011
Published 12 August 2011
Online at stacks.iop.org/PMB/56/N175

Abstract
Monte Carlo (MC) methods are the gold standard for modeling photon and electron transport in a heterogeneous medium; however, their computational cost prohibits their routine use in the clinic. Cloud computing, wherein computing resources are allocated on-demand from a third party, is a new approach for high performance computing and is implemented to perform ultra-fast MC calculation in radiation therapy. We deployed the EGS5 MC package in a commercial cloud environment. Launched from a single local computer with Internet access, a Python script allocates a remote virtual cluster. A handshaking protocol designates master and worker nodes. The EGS5 binaries and the simulation data are initially loaded onto the master node. The simulation is then distributed among independent worker nodes via the message passing interface, and the results aggregated on the local computer for display and data analysis. The described approach is evaluated for pencil beams and broad beams of high-energy electrons and photons. The output of cloud-based MC simulation is identical to that produced by single-threaded implementation. For 1 million electrons, a simulation that takes 2.58 h on a local computer can be executed in 3.3 min on the cloud with 100 nodes, a 47$\times$ speed-up. Simulation time scales inversely with the number of parallel nodes. The parallelization overhead is also negligible for large simulations. Cloud computing represents one of the most important recent advances in supercomputing technology and provides a promising platform for substantially improved MC simulation. In addition to the significant speed up, cloud computing builds a layer of abstraction for high performance parallel

[*] This work was presented in part at the 2010 Annual Meeting of the American Association of Physicists in Medicine (AAPM), Philadelphia, PA.
[3] Author to whom any correspondence should be addressed.

computing, which may change the way dose calculations are performed and radiation treatment plans are completed.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

Monte Carlo (MC) simulation is an accurate model for computing radiation dose distributions and is becoming an indispensable tool for treatment planning and verification (Bednarz and Xu 2009, Bush *et al* 2011, Bush *et al* 2010, DeMarco *et al* 1999, Hirayama *et al* 2005, Li *et al* 2011, Wang and Rogers 2010). Its routine use, however, is hampered because of the enormous computational time required to achieve adequate statistical significance in simulating over billions of particles. Fast MC simulation tools have been the subject of numerous research papers, but its ideal platform remains elusive. In particular, MC in a heterogeneous computing platform with GPU has been thought after for simulation of high-energy electrons (Bakhtiari *et al* 2010, Hissoiny and Ozell 2011, Jia *et al* 2010, Pratx and Xing 2011). While MC methods are often considered as ‘embarrassingly parallel’, efficient MC implementation on the GPU is complicated by the fact that SIMD (single instruction multiple data) architectures cannot compute diverging particle histories in parallel (Pratx and Xing 2011) and thus the resulting speed-up of GPU-based MC calculation is limited (Jia *et al* 2010). Several solutions have been proposed, such as creating a global queue of particles to be processed placing newly created secondary particles in such queue, and processing electrons and photons sequentially rather than concurrently. But this requires rewriting the simulation codes to adapt the GPU architecture and makes general use of the programs problematic.

Another strategy to speed up the MC calculation is to use computer clusters, in which particles histories are calculated independently in parallel, provided that the random number generators are initialized carefully. Significant acceleration has been accomplished using computer clusters (Badalabd and Sempau 2006, Tyagi *et al* 2004). In reality, the management of clustered computers and data flow is complicated and time consuming. Robustness of the calculation depends on the functionality of each individual computer. Massively parallel computing resources are often lacking at medical institutions because of high costs of cluster computers, energy, maintenance and storage of a dedicated cluster. Furthermore, the lack of elastic scaling often results in under- or over-utilization of computer resources.

We present a cloud computing solution for ultra-fast MC calculations by taking advantage of an emerging supercomputing technology referred to as cloud computing. Cloud computing is harnessed from the recent development of virtualization technology. Briefly, virtualization software abstracts the underlying hardware architectures (such as servers, storage and networking) and allows the system to behave like a single computer, but with flexible specification of details such as the number of processors, memory, disk size and operating system. Furthermore, cloud computing, or the on-demand virtual computers, has been made conveniently accessible by a number of third-party service providers (such as IBM, Google, Oracle, Microsoft or Amazon). Cloud computing providers offer a wide variety of services, such as software, data storage facilities, virtual web servers and scalable clusters of virtual nodes. Not linked to the hardware, the nodes and memories are allocated on demand. The computer resources can also be scaled in real-time according to demand, without having to physically power on or off computers. A detailed review of cloud computing in various biomedical applications can be found in Bateman and Wood (2009), Dudley *et al* (2010), Fox (2011), Schadt *et al* (2010). In the following, we first provide some technical details

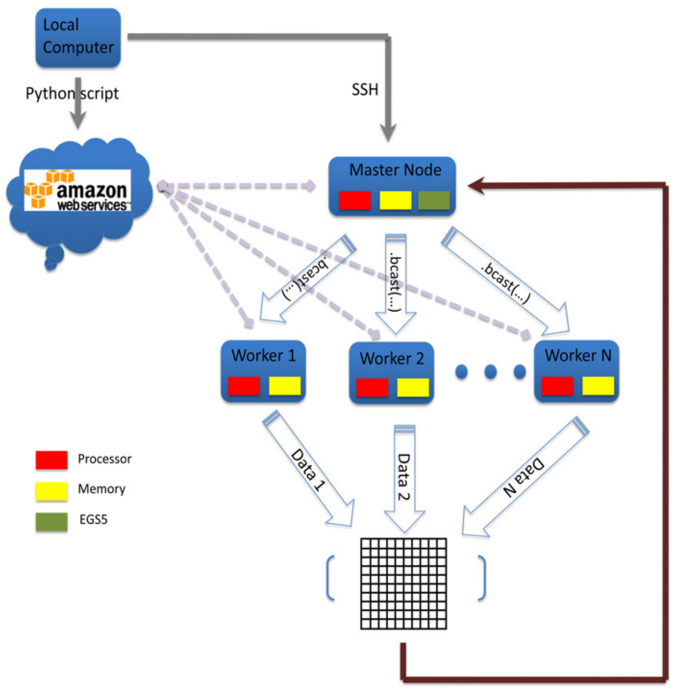

Figure 1. Schematic of a high level view of the cloud-based MC computing. A Python script is executed on a local computer to allocate via the Internet one master node and a chosen number of worker nodes. Each worker node simulates a chosen number of particles independently and the final result is transferred to the local client computer.

on our implementation of EGS5 MC simulation in the Amazon Elastic Compute Cloud (EC2) platform. EGS5 has already been demonstrated to be a valuable tool to simulate medical linacs and radiation transport (Hirayama *et al* 2005). Our implementation is then benchmarked against the original MC calculation on the CPU. The general reference drawn from our initial experience is that cloud-based environment is able to provide the scalability and performance urgently needed to meet the increasing demand for high performance MC simulation. MC simulation on a cloud architecture could bring to medical physics a fast, reliable and economically sensible solution to address various unmet clinical needs.

2. Methods

Figure 1 shows a schematic drawing of the proposed cloud-based MC computing. Distributed MC implementation was deployed in a cloud environment using the message passing interface (MPI), an API for parallel computing and a dominant model used in high performance computing (Gropp *et al* 1999, Pacheco 1996). MPI has a core of library routines for writing message-passing programs in Fortran, C, C++ or Python. An MPI program consists of autonomous processes, each executing code in its own local memory. This distributed memory model enables a much higher memory and cache which translates to higher performance MC simulation. The processes communicate via calls to MPI communication primitive routines. For MC simulations, we focus on MPI's collective communications, the capability to communicate between multiple processes simultaneously, including broadcasting, spreading and collecting data.

A useful feature of MPI is that the code can be embedded in any serial program starting with a header file that is included in the beginning of a program. Subsequent parallel code can be inserted anywhere in the original serial program as function calls to an external MPI library. The ability that MPI can be integrated into any serial code makes it extremely portable to any open source implementation. In our application, we used MPI4py, a Python version of MPI (Python 2011). We embedded parallel code directly inside a Python file consisting of serial EGS5 code. This code is executed on different machines in the cloud. Inside the parallel segment, we make calls to existing functions already written for the serial program. The calls will be dependent on the process rank, which is a unique processor ID assigned by MPI and becomes distinct once the file is executed on different machines.

In our approach, the user executes a Python script on a local computer to allocate via the Internet one master node and a chosen number of worker nodes on Amazon's EC2. The nodes are launched using Amazon AWS connection sockets, provided by Amazon and imported into Python script. Once the nodes are launched, a second script pings the status of the launched nodes and parses the returned response into instance name, public DNS address and machine status. A network file system (NFS) is set to mount a disk partition to the compute nodes from the master node. NFS provides an intelligent file sharing mechanism so that EGS5 only has to be uploaded once to the master node. EGS5 input files are then uploaded to the master node from the local computer. These files contain, for instance, the geometry of the simulated phantom, beam parameters, materials and subroutine for scoring results.

The Python file consisting of MPI code embedded inside EGS5 serial code is executed via a single command line call [using a command *mpiexec*()]. The file will prompt the user for parameters such as the number, type and energy of the simulated particles. These parameters are then distributed using the collective communication MPI [*broadcast*()] to each of the compute nodes. Although each compute node is essentially executing the same Python file, it is assigned by MPI a unique process rank, from *comm.get_rank*(). This rank parameter is used to determine which particles are simulated on which process. After simulation is finished, the resulting dose distribution is returned using another collective communication call *MPI.reduce*(). The final result—that is, the summation of the partial dose distributions returned by individual nodes—is transferred to the master node and the local client computer.

To evaluate the framework, MC simulations were performed in a cubical water phantom for both electrons and photons of different energies. For electrons, we used a mono-energetic electron pencil beam of 20, 15, 12, 9 and 6 MeV. We also simulated a mono-energetic photon pencil beam of 20, 10 and 6 MeV. Percentage depth dose curves and/or beam profiles are computed for these beams. The calculation results of the EC2 are benchmarked by comparing with that obtained using a desktop personal computer (PC).

3. Results

The left and middle panels of figure 2 show percentage depth dose (PDD) curves of pencil beam electrons and photons of different energies generated from EGS5 using our cloud computing implementation. Separate simulations were performed for varying beam energy using 1 million particles. Dose deposition in a cubic phantom was stored in a discrete 3D array containing $14 \times 14 \times 14$ cubical voxels, of size 0.5 cm^3. The dose profiles at three different depths for 20 MeV electrons of a 10 cm $\times$ 10 cm field size are plotted in the right panel of figure 2. The same set of simulations was performed using a local desktop PC and the results were found to be identical when we use the same random seed, since EGS5 is deterministic once a random seed is set.

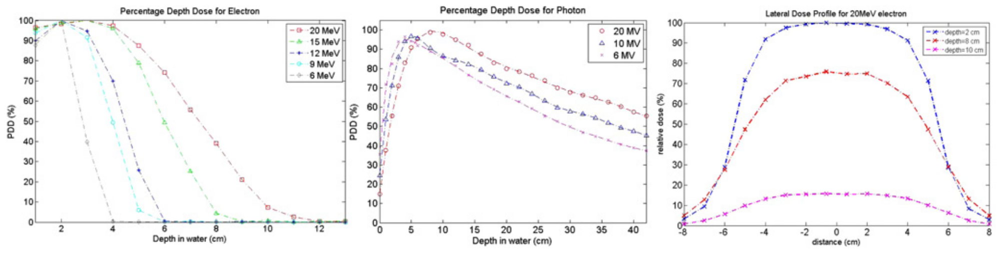

Figure 2. PDD curves for pencil beam electrons (left) and photons (middle) of varying energies. Lateral dose profiles for a 10 cm × 10 cm 20 MeV electron field at three depths are plotted in the right panel.

Table 1. Time on a CPU and 100 EC2 instances for 20 MeV electron simulations.

Number of particles	On 100 instances	On CPU	Factor of speed improvement
100	0.69 s	1.04 s	1.56×
1000	1.91 s	9.42 s	4.93×
10×10^4	2.62 s	1.56 min	35.79×
10×10^5	24.52 s	15.55 min	38.04×
10×10^6	3.30 min	2.58 h	46.90×
10×10^7	33.03 min	25.99 h	47.21×

Computational performance is the main reason for us to seek for cloud computing solution. To compare the speed of the original and the cloud computing EGS5 implementations, we fixed the energy level at 20 MeV for electron and varied the number of simulated particles. Table 1 shows a comparison of computing time of 100 EC2 nodes versus a local computer. These EC2 nodes run the Linux operating system and come with 7.5 GB of memory and two virtual cores, equivalent to four 2007 Xeon processors. The local computer has a 2.66 GHz Intel Xeon and 3 GB of memory.

Table 1 shows the speed-up factor using 100 EC2 instances versus a local computer. We achieved a 47× speed improvement in simulating 10 million particles on 100 instances. For 1 million electrons, a simulation that takes 2.58 h on a local computer can be executed in 3.3 min on the cloud. The relationship between the number of particles and the processing time for both cloud computing and CPU is linear, which confirms our theoretical expectation from MC simulations.

We also varied the number of nodes from 1 to 100 and recorded the compute time for a 10 000 particle simulation with 20 MeV electrons (figure 3). A curve of the form $y = \frac{31.33}{x^{0.987}}$, which is also plotted, fits the data very well, with an R^2 of 0.998, hence revealing the $1/\times$ scaling of the run time for one node by the number of nodes that we theoretically expect.

4. Discussion

Cloud computing allows users to perform massively parallel computation from a single computer with an Internet connection, avoiding the need for purchasing, installing and maintaining on-site hardware. Virtualization technology enables the user to treat compute resources such as data storage and compute time as commodities that can be purchased when needed. In this scheme, hardware failures have minimal impact on applications since

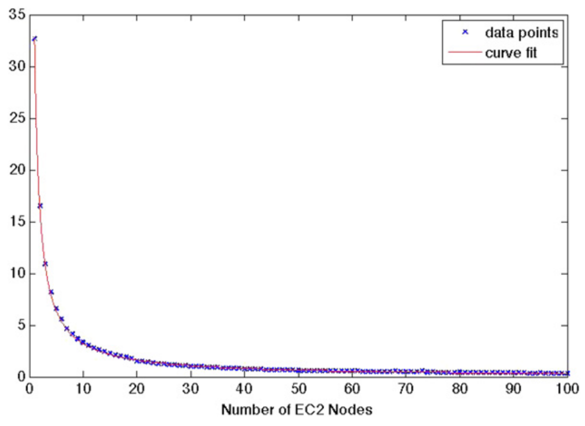

Figure 3. Time versus number of EC2 nodes for 10 000 particles, 20 MeV electrons.

storage uses redundancy mechanisms and failed nodes are seamlessly replaced with new ones. Last, cloud computing resources scale elastically with the user's demand. For instance, MC simulation requires large amounts of data storage and compute time. These resources can be allocated when needed from an almost infinite pool of resources. It is useful to emphasize that cloud computing is not yet another simple extension of the existing cluster or grid-based computing method. Instead, it represents one of the most important advances in supercomputing technology after heterogeneous computing such as GPU. Additionally, while the commercial cloud computing infrastructure made it possible for some relative naïve computer users to solve problems on supercomputers, there exists no turn-key solution for programming in a cloud environment yet. The script and MPI codes developed for the described application will be freely available upon request for research purpose.

In treatment planning, one must balance the trade-off between speed and accuracy. For this reason, many clinical applications rely on approximate dose simulation to achieve the required speed, at the cost of a lower accuracy. Cloud computing enables one to achieve both accuracy and speed, for a modest cost: the price per hour for a high-performance EC2 node is $0.38 and $0.52 for a Linux and Windows platform, respectively (Amazon 2011). One of the limitations of cloud computing is that the learning curve is still quite high, as it is for most parallel programming applications. While it is not trivial to take any MC package and deploy it in a commercial cloud computing environment without writing some additional layers of abstraction, our experience indicated that implementation of a MC simulation in a cloud computing environment is substantially easier than doing the same in a GPU platform, which requires one to completely rewrite the simulation codes to accommodate the GPU hardware configuration (Pratx and Xing 2011).

5. Conclusion

This work introduces a yet-to-be explored and much needed emerging supercomputing platform and reports our implementation experience in using the novel cloud computing platform to solve a longstanding and important problem in medical physics, i.e. how to substantially speed up the current MC calculation to a level that is clinically practical. The distributed implementation of MC simulation reported here demonstrated that cloud computing is a practical solution for computationally demanding tasks in medical physics. Substantial increase in computing speed of MC simulation was achieved using the on-demand virtual computers. By using 100 Amazon EC2 nodes, a 47× speed-up was achieved without any modification of the original EGS5 simulation code, as opposed to the GPU approach, where much of the existing MC code has to be rewritten completely. Our proof-of-concept study reveals the potential value of the emerging cloud computing infrastructure and lays the foundation for full-fledged MC applications, such as linear accelerator treatment head simulations, patient dose calculation and detector design. Finally, we emphasize that the work can be easily extended to run other MC simulation packages (such as Geant4, MCNP and Fluka) and many other computationally intensive medical physics tasks in a cloud supercomputing environment.

Acknowledgments

This project was supported in part by grants from NCI (1R01 CA133474 and 1R21 CA153587) and the NSF (0854492).

References

Amazon 2011 http://aws.amazon.com

Badalabd A and Sempau J 2006 A package of Linux scripts for the parallelization of Monte Carlo simulation *Comput. Phys. Commun.* **175** 440–50

Bakhtiari M, Malhotra H, Jones M D, Chaudhary V, Walters J P and Nazareth D 2010 Applying graphics processor units to Monte Carlo dose calculation in radiation therapy *J. Med. Phys.* **35** 120–2

Bateman A and Wood M 2009 Cloud computing *Bioinformatics* **25** 1475

Bednarz B and Xu X G 2009 Monte Carlo modeling of a 6 and 18 MV Varian Clinac medical accelerator for in-field and out-of-field dose calculations: development and validation *Phys. Med. Biol.* **54** N43–57

Bush K, Gagne I M, Zavgorodni S, Ansbacher W and Beckham W 2011 Dosimetric validation of Acuros XB with Monte Carlo methods for photon dose calculations *Med. Phys.* **38** 2208–21

Bush K, Zavgorodni S, Gagne I, Townson R, Ansbacher W and Beckham W 2010 Monte Carlo evaluation of RapidArc oropharynx treatment planning strategies for sparing of midline structures *Phys. Med. Biol.* **55** 4465–79

DeMarco J J, Solberg T D and Chetty I 1999 Monte Carlo methods for dose calculation and treatment planning: a revolution for radiotherapy *Adm. Radiol. J.* **18** 24–7

Dudley J T, Pouliot Y, Chen R, Morgan A A and Butte A J 2010 Translational bioinformatics in the cloud: an affordable alternative *Genome Med.* **2** 51

Fox A 2011 Computer science. Cloud computing–what's in it for me as a scientist? *Science* **331** 406–7

Gropp W, Lusk E and Skellum A 1999 Using MPI: portable parallel programming with the message-passing interface (Cambridge, MA: MIT Press)

Hirayama Y N H, Bielajew A F, Wilderman S J and Nelson W R 2005 The EGS5 code system *SLAC Report No 730*

Hissoiny S and Ozell B 2011 GPUMCD: a new GPU-oriented Monte Carlo dose calculation platform *Med. Phys.* **38** 754–64

Jia X, Gu X, Sempau J, Choi D, Majumdar A and Jiang S B 2010 Development of a GPU-based Monte Carlo dose calculation code for coupled electron-photon transport *Phys. Med. Biol.* **55** 3077–86

Li X, Samei E, Segars W P, Sturgeon G M, Colsher J G, Toncheva G, Yoshizumi T T and Frush D P 2011 Patient-specific radiation dose and cancer risk estimation in CT: part I. Development and validation of a Monte Carlo program *Med. Phys.* **38** 397–407

Pacheco P S 1996 Parallel programming with MPI (San Mateo, CA: Morgan Kaufmann Publishers)

Pratx G and Xing L 2011 GPU computing in medical physics: a review *Med. Phys.* **38** 2685–97

Python 2011 http://pypi.python.org/pypi/mpi4py

Schadt E E, Linderman M D, Sorenson J, Lee L and Nolan G P 2010 Computational solutions to large-scale data management and analysis *Nat. Rev. Genet.* **11** 647–57

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

Wang L L and Rogers D W 2010 Replacement correction factors for plane-parallel ion chambers in electron beams *Med. Phys.* **37** 461–5