Hybrid GATE: A GPU/CPU implementation for imaging and therapy applications

Julien Bert, H Perez Ponce, S Jan, Z El Bitar, P Gueth, V Cuplov, H Chekatt, D Benoit, D Sarrut, Y Boursier, et al.

Hybrid GATE: A GPU/CPU implementation for imaging and therapy applications

Julien Bert, Hector Perez-Ponce, Sébastien Jan, Ziad El Bitar, Pierre Gueth, Vesna Cuplov, Hocine Chekatt, Didier Benoit, David Sarrut, Yannick Boursier, David Brasse, Irène Buvat, Christian Morel, and Dimitris Visvikis

***Abstract*—Monte Carlo simulations (MCS) play a key role in medical applications. In this context GATE is a MCS platform dedicated to medical imaging and particle therapy. Yet MCS are very computationally demanding, which limits their applicability in clinical practice. Recently, graphics processing units (GPU) became, in many domains, a cost-effective solution to access high power computation. The objective of this work was to develop a GPU code targeting MCS for medical applications integrated within the GATE software. An aim was to enhance GATE computational efficiency by taking advantage of GPU architectures. We first developed a GPU framework with basic elements to run MCS for different medical applications. The implementation was based on a GPU adaptation of the Geant4 code. For each main GATE medical application, we developed a specific code from the GPU framework. Some of these GPU codes are currently being integrated in GATE as new features, and users can perform GPU computing in their GATE simulations. The acceleration factor resulting from the implementation of the tracking within the phantom on GPU was 60 for a PET simulation and 80 for a CT simulation. By using GPU architectures, we are also extending GATE to support optical imaging simulations that are heavily demanding in terms of computational resources. Radiation therapy applications currently supported by GATE V6.2 are also being adapted to run on hybrid GPU/CPU architectures.**

I. Introduction

MONTE Carlo simulations (MCS) are random sampling methods used in many domains to simulate and solve physical and mathematical problems. MCS play a key role in medical applications. In this context, GATE is a MCS platform dedicated to medical imaging and particle therapy [1]. Yet, MCS are very computationally demanding, which limits their applicability in clinical practice. The use of computer clusters can help solve this intensive computational issue, but has not yet been widely adopted in clinical environment.

Recently, graphics processing units (GPU) became, in many domains [2], a cost-effective solution to access high power computation. This architecture is able to deliver computation power comparable to that of a small cluster on a single conventional workstation. Some previous studies [3]–[5] have shown that the use of GPUs in MCS is promising to significantly reduce execution times. However the proposed GPU codes were optimized and developed for specific applications. Such codes are proofs of concept but not user-oriented, which limits the simulation possibilities.

The objective of this work was to develop a GPU code targeting MCS for medical applications integrated within the GATE software. An aim was to enhance GATE computational efficiency by taking advantage of GPU architectures. The purpose of our work is to integrate GPU modules within the GATE simulation platform in a hybrid manner. At different point of the simulation it will be possible to track particles alternatively on GPU or CPU. The proposed implementation allows the use of CPU and GPU modules without limiting the overall capabilities of the GATE platform.

J. Bert and D. Visvikis are with the LaTIM, UMR1101 INSERM, CHRU Brest, France (e-mail: julien.bert@univ-brest.fr).

H. Perez-Ponce, Y. Boursier and C. Morel are with the CPPM, Aix-Marseille Université, CNRS/IN2P3, Marseille, France.

Z. El Bitar, H. Chekatt and D. Brasse are with the Institut Pluridisciplinaire Hubert Curien, UMR 7178 - CNRS/IN2P3, Strasbourg, France.

P. Gueth and D. Sarrut are with the Université de Lyon, CREATIS, CNRS UMR5220, INSERM U630, INSA-Lyon, Université Lyon 1, Centre Léon Bérard, France.

V. Cuplov and S. Jan are with the DSV/I2BM/SHFJ, Commissariat à l'Energie Atomique, Orsay, France.

D. Benoit and I. Buvat are with the IMNC-UMR 8165 CNRS-Paris 7 and Paris 11 Universities, Orsay, France.

II. Materials and methods

A. Global strategy

A GPU is composed of thousand of threads, each one representing a data unit, in other words a piece of data to be processed. All threads will execute the same program code, called kernel, in parallel on the different streaming processors, representing individual processor units. Recent GPU architectures contain more than one thousand processors. The paradigm we have chosen for the MCS GPU implementation is the use of one thread per particle, i.e. a thread handles a given particle from its birth to its death. Using thousands of processing units, thousands of particles will be simulated in parallel by executing the same code on the GPU.

Starting from this basic paradigm, a GPU pipeline was developed for the MCS. As particles run simultaneously we need to keep in memory every particles by storing them inside a buffer. This buffer is then presented to a first kernel to perform the particles' navigation inside a voxelized geometry. Consequently all particles are moved by one geometrical step defined by the next voxel boundary, or one physical step determined by the next physics interaction. Particles reaching the phantom boundary will be stopped. Then a second kernel is used to apply the appropriate physics for each particle. For example some particles will be scattered or absorbed according to their states. At the end of this iteration on all particles, if there are still particles to process within the buffer, the navigation and interaction steps are repeated until all particles are processed. All basic functions used within a complete

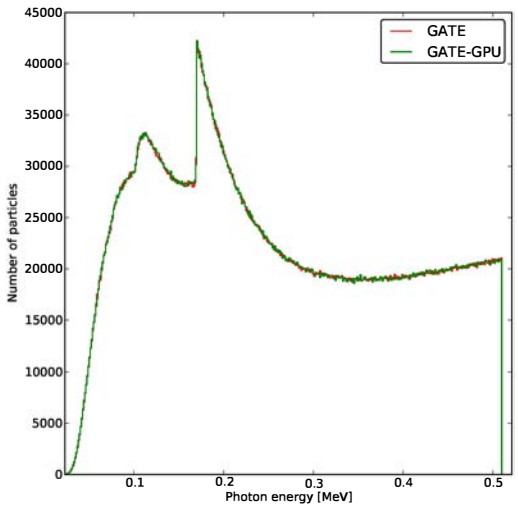

Fig. 1. Energy distribution for scattered photons from emission imaging simulation.

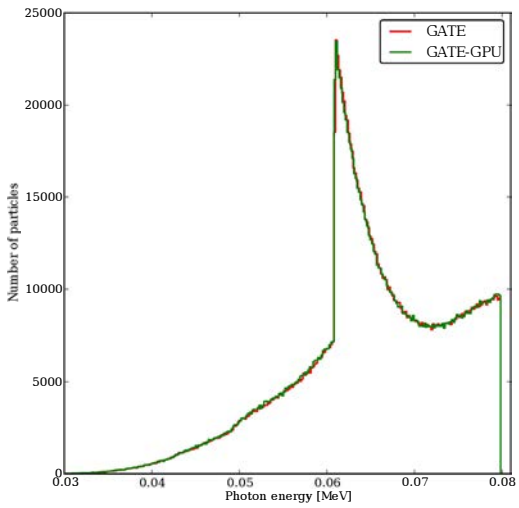

Fig. 2. Energy distribution for scattered photons from transmission imaging simulation.

MCS were developed within a GPU framework [6] using the parallel computing platform CUDA created by NVIDIA. The implementation was based on a GPU adaptation of the Geant4 code [7]. This GPU framework was subsequently assessed against Geant4.

For each main GATE medical application, we developed a specific code from the GPU framework. According to the specificities of the application, we implemented the relevant physics effects on the GPU. Some of these GPU codes are currently being integrated in GATE as new features, and users can perform GPU computing in their GATE simulations. The GATE application supporting GPU makes it possible to track particles alternatively on the GPU modules and on the regular GATE CPU components in a hybrid manner, so that GPU modules can be turned on and off. One challenge was the management of a hybrid implementation combining a sequential CPU simulation and a parallel GPU simulation. Such hybrid implementation was made possible using a buffering system. For example, a simple GATE simulation could consist of a CPU source, our GPU particle tracker within a voxelized phantom and a CPU detector. The source produces particles sequentially inside the GATE workspace. Any particle crossing the phantom boundary is stored on the GPU buffer. When the buffer is full, the GPU simulation is triggered. Subsequently tracking within the voxelized object is performed in parallel by the GPU for all particles. Then each particle simulated by the GPU is passed sequentially on the standard GATE workspace in order to continue the simulation and reach or not the detector.

B. Application for emission imaging

The GPU framework has been used to perform a positron emission tomography (PET) simulation in GATE. The GPU code only manages the particle tracking inside voxelized phantoms. The photon transportation in the detector and the modelling of the detector response were performed on the usual CPU GATE code. The GPU code for PET simulation was integrated within GATE as a new module named GPU-PET that provides new input (i.e. photons) to the regular GATE engine once the photons exit the voxelized volume.

The validation of the hybrid code was performed by simulating the same PET acquisition with and without the GPU-PET module. An NCAT voxelized phantom and a Philips GEMINI PET system model [8] were used. An activity map emitting pairs of gamma rays of 511keV and showing a lung tumor with a contrast of 5:1 was defined. The total activity simulated was 28.7MBq. The simulation was performed by using photoelectric (PE) and Compton scattering processes provided by the standard model and a fictitious tracking [9], [10]. Both simulations modeled a 10 minutes acquisition. We measured the run time to track particles within the voxelized phantom. Finally all particles escaping the phantom were stored in a phase-space file. Each simulation was performed with a NVIDIA GTX580 GPU (512 cores - 1.23GHz) and an Intel Core i7 CPU (3.4GHz).

C. Application for transmission imaging

We also adapted the GPU framework for a transmission imaging simulation. This benchmark was close to a cone beam CT geometry, imaging a single phantom projection. The GPU handles only the photon transportation within the voxelized phantom. This specific GPU code was incorporated into GATE as a new feature called GPU-CT.

To validate the GPU-CT implementation, we compared two GATE simulations, one using the GPU-CT code, and the other using only the CPU GATE code. A mono-energetic cone beam photon source at 80keV and a voxelized head and neck phantom were used. A flat panel detector was used to count the number of detected particles. The distance between the source and the phantom was 100cm when it was 20cm for the flat

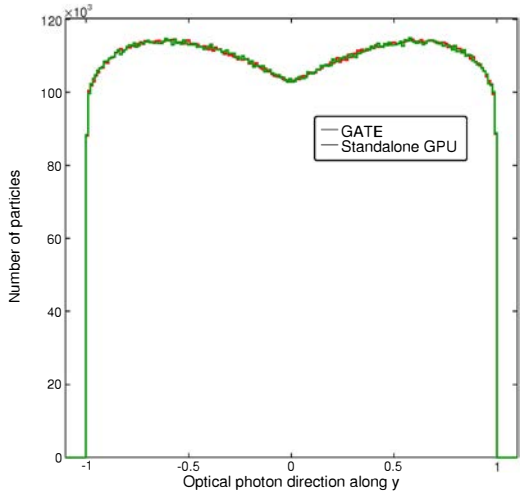

Fig. 3. Simulation of the optical photon direction along the y-axis for the Mie scattering.

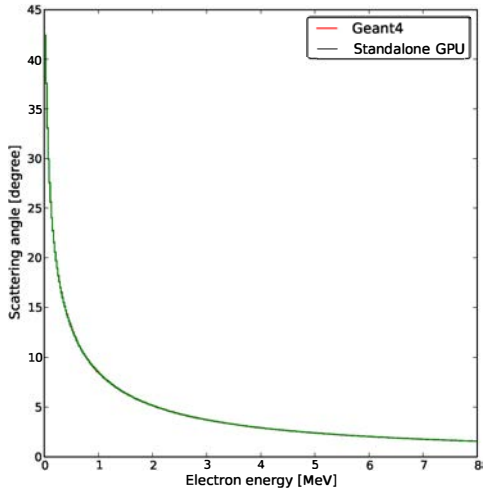

Fig. 4. Simulation of the electron scattering angle as a function of its energy for the electron ionisation effect.

panel detector. The PE and Compton scattering effects were modeled. The particle tracking within the voxelized geometry on GPU was performed by a so-called "regular" navigator based on the one provided by Geant4. A total of 500.10^6 photons were generated for each simulation. We measured the run time to track particles within the voxelized phantom. Finally all particles escaping the phantom were stored in a phase-space file. Each simulation was performed with the same graphical card and CPU as the ones used for the PET simulation.

D. Application for optical imaging

Optical imaging is an efficient and low-cost imaging technique enabling real time study of biological processes. Several physics processes occur during the optical photon propagation in biological tissues: absorption, scattering, transmission and reflection at tissue boundaries. We extended the standalone GPU framework code by adding the Mie scattering process provided by Geant4. The GPU implementation was validated against GATE simulations running only on CPU.

The simulation set-up used for validation consisted of a voxelized phantom made of a water box of $100\times100\times100$ 4mm^3 voxels. The water anisotropy and Mie scattering length were set to 0.62 and 6mm respectively. We simulated an isotropic source of 20.10^6 optical photons of 6eV located at the center of the water box. After tracking all particles up to the water box boundary, the particles were stored in a phase-space file.

E. Application for radiotherapy

The targeted radiotherapy applications were the calculation of the dose distribution induced by photon or electron beams in voxelized phantoms. For these applications, the GPU framework is being extended to support new physics effects, including electron ionisation, multiple scattering and Bremsstrahlung. Secondary particles also have to be taken into account. This GPU code is still under development, but a standalone GPU code for the electron ionization effect provided by Geant4 has already been implemented.

A preliminary validation of the GPU code against Geant4 was achieved by simulating a voxelized water box ($100\times100\times100$ 1cm^3 voxels) including an isotropic electron source (uniform energy ranging from 0 to 8MeV) at the center. We simulated 20.10^6 particles both for the GPU and Geant4 codes. Information about the particle scattering angle was stored during the simulation.

III. Results

For the PET results, the run time for tracking one million particles inside the voxelized volume on standard GATE was 75.4s against only 1.23s for hybrid GATE (acceleration factor of 61.3). This suggests that the use of GPU in GATE has the potential to convert hours of simulations into minutes. From the saved phase-space files given by both simulations, we compared the scattered photon energy distribution by plotting the number of particles as a function of the photon energy (Fig. 1), showing a good agreement between standard and hybrid GATE implementations.

Results from transmission imaging simulations led to similar conclusions. The run time for tracking one million particles wihtin the voxelized phantom on standard GATE was 89.4s and only 1.16s for hybrid GATE. The GPU module was 77.1 times faster that the CPU one. A comparison of the plotted scattered energy distribution given by both phase-space files also showed a very good agreement in Fig. 2.

For the optical imaging application, Fig. 3 shows the optical photon direction along the y-axis obtained with GATE and the standalone GPU implementation of the Mie scattering process. For the radiotherapy application, the distribution of the scattering angle as a function of the electron energy is shown in Fig. 4 for the GPU implementation and for the

pure CPU Geant4 code. These two figures also confirm the accuracy of the GPU implementation of the corresponding physics process.

IV. Conclusion and further work

The aim of our work is to develop a GPU code for the main medical applications available in the GATE software. In this work, we described GATE emission and transmission tomography applications taking advantage of a GPU architecture to enhance their computational efficiency. By using GPU architectures, we are also extending GATE to support optical imaging simulations that are heavily demanding in terms of computational resources. Radiation therapy applications currently supported by GATE V6.2 are also being adapted to run on hybrid GPU/CPU architectures. Further work will include the implementation of additional GATE modules running on GPU and the complete validation of a hybrid GPU-CPU GATE version.

Acknowledgment

This work was funded by the French National Research Agency through hGATE project (ANR-09-COSI-004-01).

References

[1] S. Jan et al., "Gate v6: a major enhancement of the gate simulation platform enabling modelling of ct and radiotherapy," *Physics in medicine and biology*, vol. 56, pp. 881–901, 2011.

[2] J. Nickolls and W. J. Dally, "The gpu computing era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, Mar/Apr. 2010.

[3] B. Toth and M. Magdics, "Monte carlo radiative transport on the gpu," in *Fifth Hungarian Conference on Computer Graphics and Geometry*, 2010, pp. 1–8.

[4] S. Hissoiny, B. Ozell, H. Bouchard, and P. Desprs, "Gpumcd: A new gpu-oriented monte carlo dose calculation platform," *Medical Physics*, vol. 38, no. 2, pp. 754–764, 2011.

[5] L. Jahnke, J. Fleckenstein, F. Wenz, and J. Hesser, "Gmc: a gpu implementation of a monte carlo dose calculation based on geant4," *Physics in medicine and biology*, vol. 57, pp. 1217–1229, 2012.

[6] H. Perez-Ponce, Z. E. Bitar, Y. Boursier, D. Vintache, A. Bonissent, C. Morel, D. Brasse, D. Visvikis, and J. Bert, "Implementing geant4 on gpu for medical applications," in *IEEE Nuclear Science Symposium and Medical Imaging Conference*, 2011, pp. 2703–2707.

[7] J. Allison et al., "Geant4 developments and applications," *IEEE Transactions on Nuclear Science*, vol. 53, no. 1, pp. 270–278, 2006.

[8] F. Lamare, A. Turzo, Y. Bizais, C. C. L. Rest, and D. Visvikis, "Validation of a monte carlo simulation of the philips allegro/gemini pet systems using gate," *Physics in medicine and biology*, vol. 51, pp. 943–962, 2006.

[9] L. L. Carter, E. D. Cashwell, and W. M. Taylor, "Monte carlo sampling with continuously varying cross sections along flight paths," *Nuclear Sience and Engineering*, vol. 48, pp. 403–411, 1972.

[10] N. S. Rehfeld, S. Stute, J. Apostolakis, and M. Soret, "Introduction improved voxel navigation and fictitious interaction tracking in gate for enhanced efficiency," *Physics in medicine and biology*, vol. 54, pp. 2163–2178, 2009.