Photon Optimization with GPU and Multi-Core CPU; What are the issues?

Arezoo Modiri, PhD

Outline

• Parallelization
  ▪ CPUs/Clusters/Cloud/GPUs
  ▪ Data management

• Computation-Intensive Applications in Photon Radiotherapy
  ▪ Dose calculation
  ▪ Image registration/reconstruction
  ▪ Robustness analysis
  ▪ Higher-dimensional inverse planning

• Through an Example (4D IMRT Inverse Planning)
  ▪ Hardware configuration
  ▪ Factors impacting process speed

Disclosure

This work was supported in part by the National Cancer Institute (R01CA169102) and Varian Medical Systems.
Why is parallelization important?

- Radiotherapy applications use large data sets and/or complex numerical algorithms.
- Are desired to be solved in a timely fashion.
- Are sometimes desired to be solved in minutes or even in (near) real time, such as on-line adaptive radiation therapy (ART).

Why is parallelization important?

- Radiotherapy applications use large data sets and/or complex numerical algorithms.
- Are desired to be solved in a timely fashion.
- Are sometimes desired to be solved in minutes or even in (near) real time, such as on-line adaptive radiation therapy (ART).

Solutions for Speeding Up Processes

- Using devices with higher clock speed (we are hitting a technological limit)
- Using devices supporting parallel processing (multi-core CPUs and GPUs)

Solutions for Speeding Up Processes

- Using devices with higher clock speed (we are hitting a technological limit)
- Using devices supporting parallel processing (multi-core CPUs and GPUs)

Solutions for Speeding Up Processes

- Managing data for parallel processing
  - The size of the data can be large. Yet, data are usually parallelization-friendly, in that the entire task can be naturally broken down to small operations at pixel/voxel/beam/beamlet/aperture/etc.-level.
  - Most works use single-precision float point data type.
  - Down sampling, reducing calculation volume and sparsification can be used to manage data.
Solutions for Speeding Up Processes

• Using devices with higher clock speed (we are hitting a technological limit)
• Using devices supporting parallel processing (multi-core CPUs and GPUs)

• Managing data for parallel processing
  – The size of the data can be large. Yet, data are usually parallelization-friendly, in that the entire task can be naturally broken down to small operations at pixel/voxel/beam/aperture/etc. level.
  – Most works use single-precision float point data type.
  – Down sampling, reducing calculation volume and sparsification can be used to manage data.
• Intermediate-size data: A GPU solution
• Large-size data: A CPU solution

GPUs versus CPUs

Clusters

• Expensive (setup and maintenance)
• Performance dependency on number of users
• Citrix is an example
Cloud-based Clusters
Outsourcing computation resources to a 3rd party company (Amazon, google, etc.)
- Internet browser enabling a user to view DICOM-RT file.
- Performs computing tasks (registration, segmentation, treatment planning, dose calculation)
- Data base
Need to pay per hour

Reviews of GPU-based Computation in Radiotherapy
GPU Computing in Medical Physics, Lei Xing et al., Med. Phys. 2011
GPU-based high-performance computing for radiation therapy, Xun Jia et al., PMB 2014

Computationally Intense Radiotherapy Applications
- Dose calculation
- Image registration/reconstruction
- Plan optimization
- Robustness analysis
Computationally Intense Radiotherapy Applications

An Example

ECHO (Expedited Constrained Hierarchical Optimization)

- Computational time 1 to 4 hours
- Express the clinical criteria as hard constraints
- Prioritize the clinical objectives and optimize them in order

Depending on data size, registration may take less or more time compared to plan optimization (from our group's study).

Monte Carlo Dose Calculation

- GPU-based MC project at UT Southwestern
  - Particle types: photon, electron, proton, carbon ion, free radical...
  - Clinical applications: external beam therapy, brachytherapy
  - Energy ranges: eV → keV → MeV → GeV
  - Spatial scales: nm (DNA level) → m (human level)
Clinical Application of MC Dose Calculation

- Dose calculation
- Including imaging dose in optimization
- Treatment monitoring/verification

Dose Calculation

Efficient MC Implementation

- Instead of calculating dose deposition matrices for all beamlets using MC prior to optimization, dose calculation is performed inside optimization loop but number of particles for MC is optimized.

Offline vs Online

CT was resampled to 128 × 128 × 86 voxels.

- The computation time including both MC dose calculations and plan optimizations was reduced by a factor of 4.4, from 494 to 113 s, using only one GPU card.

Dose Calculation

Hardware-Independent Implementation

- In terms of efficiency, goMC was ~4–16% slower than gDPM when running on the same NVIDIA TITAN card for all the cases tested, due to both the different electron transport models and the different development environments.
- AMD GPU cards are faster for OpenCL applications.
It is quite straightforward to port an existing CPU algorithm onto GPU and achieve acceleration to a certain degree. It is, nonetheless, quite challenging to write a high-efficiency code that fully exploit the potential of a GPU.

**Dose Calculation**

**Hardware-Independent Implementation**

It is quite straightforward to port an existing CPU algorithm onto GPU and achieve acceleration to a certain degree. It is, nonetheless, quite challenging to write a high-efficiency code that fully exploit the potential of a GPU.

**MC Imaging Photon-Electron Simulation**

- GPU implementation of the photon transport mechanism of EGSnrc
- Speedups of 20 to 40 times for 64^3 to 256^3 voxels were observed

**Thread divergence control**

**MC Particle Transport Simulation**

- Using parametrized geometry, the computational time ranged in 1.75–2.03 times the voxelized geometry for coupled photon/electron transport depending on the voxel dimension of the auxiliary index array, and in 0.69–1.23 times for photon only transport.

**Algorithmic solutions**
Robustness Analysis

- Determining the geometric uncertainties effects on the quality of the RT plans is computationally expensive and demands high performance computation capabilities.

- An in-house radiation therapy robustness analyzer (RTRA)
  - Simulates uncertainties due to:
    - Daily patient setup error
    - Deformable body motion
    - Delineation uncertainties

Higher-Dimensional Inverse Planning

- IMRT (Ziegenhein et al., PMB 2013; Men et al., PMB 2009)
- 4D (Nohadani et al., PMB 2010; Suh et al., PMB 2009)
- 4\pi (Chiu et al., Med. Phys. 2016; Dong et al., RED 2012)

An Example: 4D IMRT Inverse Planning

The pipeline for our work consisted of:
(i) creating treatment plans for each phase in Eclipse 13.6 TPS,
(ii) exporting dose-deposition matrices for all (tens of thousands) apertures,
(iii) optimizing aperture MU weights using GPU-based in-house optimization.

For 4D dose summation, we used a GPU-enabled deformable image package (Elastix).
An Example: 4D IMRT Inverse Planning

Parallelized over phases

Deformable image registration:
- Open source packages
- Commercial TPS
- In-house tools

Search agents in particle swarm optimization

Parallelized over particles

Reading datasets in sparse matrices

Phase 1
Scaling sparse matrices
Phase N
Scaling sparse matrices

Updating weights

Particle P
Calculating objective function

Finding Global Best
Finding Personal Best

No Termination criterion met?

Yes

Iterating over phases

Updating velocities

Creating individual-phase treatment plans in Eclipse TPS

Creating dose matrices per aperture using ESAPI

Exporting CT scans, structure masks and dose matrices using ESAPI

Initializing PSO particle positions and velocities

In-house Optimization Engine

In Eclipse
On CPU & GPU
On GPU

Particle P
Running partitioning operator & Creating summed matrix over phases

Eclipse
On CPU & GPU
On GPU
An Example: 4D IMRT Inverse Planning

Our implementation was hardware dependent.

12GB of memory available per GPU card
Non-uniform memory access (NUMA) structure
8-core CPUs, 256GB RAM

8/1/2017

An Example: 4D IMRT Inverse Planning

Dose matrix size
Number of particles

DIR time
PSO time
QPS time

We generally use 25-30 iterations.
An Example: 4D IMRT Inverse Planning

- More details on this study will be presented at Thursday, Session # TH-CD-205-4
  GPU-accelerated Higher Dimensional Inverse Planning, Hagan et al., University of Maryland

Other Applications

- Real-time Monte Carlo based Treatment Dose Reconstruction and Monitoring
- Dose- and Volumetrically targeted adaptive-planning
  Algorithms for micro- (small operations in parallel) and macro- (large operations) parallelization being designed
- Biological endpoint calculation using Monte Carlo
- Radiomics and artificial intelligence
  Problematic lymph node identification
  Organ-at-risk labeling given contours

Conclusion

GPU implementation has enabled various radiotherapy applications being processed in minutes or even seconds.

**Data size** is an important factor in choosing hardware configuration.

**Optimal number of GPUs** is not necessarily equal to maximum number of GPUs available.

The implementation technique and process time are **hardware dependent**.

The choice and design of **algorithms** are important in parallelization and avoidance of thread divergence.
An Example: 4D IMRT Inverse Planning

Backup Slides

Questions?

Thank you.

Arezoo Modiri
amodiri@som.umaryland.edu
Department of Radiation Oncology
University of Maryland, Baltimore

Hagan et al., University of Maryland, NIH (R01CA169102) and Varian Medical Systems.

PTV: 144 cc
Target motion: 1.5 cm
To conserve GPU memory, we used the Compressed Column Row sparsification (10:1 compression ratio). An open-source deformable image registration (Elastix), was employed for dose summation. To avoid deforming tens of thousands of dose matrices, we applied deformation vector fields, calculated prior to optimization, to summed dose matrices inside iteration loop. For evaluation, several 4D-IMRT planning tests were performed on patient data, considering 10 phases, 9 beams, 166 apertures (14940 variables).

A typical 10 phase, 200 particle study would equate to 2000 DIR operations in parallel. For a typical patient with each dose matrix being 18.7 MB in size, this would equate to 37.4 GB of dedicated GPU memory that would need to be allocated by elastix.

Process time increases both with number of particles and number of respiratory phases.
The optimal number of GPUs (five, in this study) is directly related to the hardware specifications of employed GPU cards.

An Example: 4D IMRT Inverse Planning

METHODS

Our implementation is distinct from existing 4D planning applications in commercial TPSs because
(i) it is not based on internal target volume generation,
(ii) it optimizes across phases and not for each phase, individually,
(iii) particle swarm optimization is used to solve an inverse plan optimization consisted of dose-volume-based objective function, and
(iv) aperture MU weights are optimized not fluence.

METHODS

The pipeline for our work consisted of
(i) creating treatment plans for each phase in Eclipse 13.6 TPS,
(ii) exporting dose-deposition matrices for all (tens of thousands) apertures,
(iii) optimizing aperture MU weights using GPU-based PSO, implemented in-house.

The optimal number of GPUs (five, in this study) is directly related to the hardware specifications of employed GPU cards.
METHODS

For 4D dose summatation, we used a deformable image package (Elastix). Due to GPU memory limitations, we needed to use the data in chunks and spread the tasks between GPUs and CPUs.

GPU versus CPU

- High computational power, small size, low maintenance cost
- Single instruction multiple data

<table>
<thead>
<tr>
<th>SPECIFICATIONS</th>
<th>GPU</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Memory</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>Cost</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 6: Comparing high-end products of CPU and GPU.

<table>
<thead>
<tr>
<th></th>
<th>Cost</th>
<th>Density</th>
<th>Inverse Plan Optimization</th>
<th>Inclined Plan Optimization</th>
<th>CPU Only</th>
<th>GPU Only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6750</td>
<td>2.4</td>
<td>1.2</td>
<td>16.6</td>
<td>2.2</td>
<td>24.4</td>
</tr>
<tr>
<td></td>
<td>320</td>
<td>6.5</td>
<td>6.4</td>
<td>52.7</td>
<td>216</td>
<td>174</td>
</tr>
<tr>
<td></td>
<td>320</td>
<td>6.5</td>
<td>6.4</td>
<td>52.7</td>
<td>216</td>
<td>174</td>
</tr>
<tr>
<td></td>
<td>320</td>
<td>6.5</td>
<td>6.4</td>
<td>52.7</td>
<td>216</td>
<td>174</td>
</tr>
<tr>
<td></td>
<td>320</td>
<td>6.5</td>
<td>6.4</td>
<td>52.7</td>
<td>216</td>
<td>174</td>
</tr>
<tr>
<td></td>
<td>320</td>
<td>6.5</td>
<td>6.4</td>
<td>52.7</td>
<td>216</td>
<td>174</td>
</tr>
</tbody>
</table>

Outline

Why is parallelization important?
- Dose calculation
- Inverse plan optimization
- Offline processes versus on-line/real-time/on-the-fly processes
- Fluence optimization versus aperture weight optimization
- Dealing with large number of variables in IMRT, ARC treatment planning or in 4D and non-uniform fractionation treatment planning

What is the impact of data size?
- GPU memory
- Downsampling versus keeping original data size
- Specification
- Computationally expensive processes, e.g., deformable image registration
- Staying compatible with existing treatment planning systems

Solver and algorithm matters.
- Dealing with non-convexity: DVH-based goals, BED-based goals
- Using global versus Local optimization
Dose Calculation

A new Monte Carlo-based dose calculation approach for intensity modulated radiation therapy (IMRT) has been developed within the University of Maryland’s Department of Radiation Oncology. This approach leverages the high computational speed of Graphics Processing Units (GPUs) to perform Monte Carlo simulations. The method is designed to be highly accurate while maintaining low computational requirements.

The approach involves the following steps:

1. **Simulation**: Simulate the interaction of particles with the tissue.
2. **Dose Calculation**: Calculate the dose distribution across the treatment volume.
3. **Optimization**: Optimize the treatment plan based on the dose distribution.

The key advantage of this method is its ability to perform high-speed simulations on GPUs, which can significantly reduce the time required for dose calculations and plan optimizations. This capability is particularly beneficial in clinical settings where rapid decision-making is crucial.

The method has been validated using a variety of beam types and treatment scenarios, demonstrating high accuracy and efficiency. Further research is ongoing to expand the scope of this method and to integrate it into clinical workflows more effectively.
Dose Calculation

An ultra-fast Monte Carlo dose engine for High-dose-rate brachytherapy

Monte Carlo-based particle transport has been widely used in radiation therapy to calculate dose distributions. However, the computation time for Monte Carlo simulations on a graphics processing unit (GPU) can be prohibitive, especially for complex geometries. To address this, we developed a method to incorporate parameterized geometry into a Monte Carlo particle transport simulation on a GPU.

Parameterized geometry allows for efficient representation of complex shapes. In our method, we developed a strategy to use an auxiliary index array to reduce the frequency of geometry calculations and improve efficiency. We also developed a method to store parameterized geometry data in GPU's shared memory, which further enhances performance.

We integrated our method into two previously developed GPU-based Monte Carlo packages and tested it in two example problems: (1) low energy photon transport simulation in a brachytherapy cancer case with 22 dwell locations, and (2) MeV coupled photon/electron transport in Varian Mammosite. Local point dose differences were within 2%. In a water phantom, the calculated isodose curves agreed well with those calculated in the corresponding Geant4 calculations for radial distances from 1 cm to 20 cm. The anisotropy functions were within 1% for radial distances from 1 cm to 20 cm except for polar angles larger than 173°. Local point dose differences were within 2%. In a mammography breast case with 22 dwell locations, gBMC and gPbMC showed very good agreement. The computation times for both problems were significantly reduced compared to the non-optimized GPU-based Monte Carlo packages. Our method can significantly improve the efficiency of Monte Carlo particle transport simulations on a GPU for complex geometries.
MLC Trajectory Optimization

- Graph optimization to generate efficient dynamic trajectories for delivery while maximizing the angular fluence through all PTV voxels.
- 3D dose optimization is performed for trajectories using a commercial TPS progressive resolution optimizer.

Node (Control Point) - Deliverable trajectory - Current best trajectory