Development of GPU-based Monte Carlo Simulation Packages for Radiotherapy

Xun Jia
xunjia@ucsd.edu
5/23/11
Outline

- Introduction to GPU
- gDPM
  - Motivations
  - Approaches
  - Results
- gCTD/gMCDRR
  - Motivations
  - Approaches
  - Results
- Conclusions
Monte Carlo on GPU

- Speed up MC simulations for radiotherapy on GPU
- Graphics Processing Unit
  - Turn your PC into a supercomputer
    - Tesla C2050
    - 448 processors
    - 575 MHz clock speed
    - 3 GB memory
    - >1Tflops single precision
## GPU details

<table>
<thead>
<tr>
<th>GPU card</th>
<th>GeForce 9500 GT</th>
<th>GeForce GTX 480</th>
<th>Tesla C1060</th>
<th>Tesla S1070</th>
<th>Tesla C2050</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price ($)</td>
<td>~50</td>
<td>~400</td>
<td>~1,400</td>
<td>~8,000</td>
<td>~2,500</td>
</tr>
<tr>
<td>Memory (GB)</td>
<td>1</td>
<td>1.5</td>
<td>4</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>Computing power (Gflops)</td>
<td>134</td>
<td>1344</td>
<td>936</td>
<td>4147</td>
<td>1288</td>
</tr>
<tr>
<td># of Processors</td>
<td>32</td>
<td>240</td>
<td>240</td>
<td>960</td>
<td>448</td>
</tr>
</tbody>
</table>
CUDA Programming

- Compute Unified Device Architecture
  - Enable us to program GPU via standard programming languages such as C
- An essential conflict between GPU architecture and MC simulation
  - Single Instruction Multiple Data (SIMD)
  - Branching problem in MC simulation
- Optimize memory usage
gDPM project

- Speed up full MC MV dose calculation using GPU
- Dose Planning Method
  - Designed for radiotherapy simulation
  - Fast compared to other general purposed MC packages
  - Relatively simple simulation process --- easy to program

- Key idea
  - Same physics as in DPM
  - Maintain computation accuracy
  - Obtain speed up by optimizing code for GPU architecture

- Approaches
  - First rewrite DPM in C
  - Write CUDA code on GPU
gDPM v1.0

- **Method**
  - Treat each computational thread on a GPU as an independent computing unit
  - Multiple thread run simultaneously

- **Implementation**
  - Each thread keeps its own RND seed
  - Each thread tracks its own particles
  - Transfer dose deposition in all threads to a global counter at the end of GPU kernel

- **Speed-up factors of about 5.0 ~ 6.6 times have been observed**
gDPM v2.0

- Separate photon and electron transport

**Data structure**
- Current Particles
- Photon stack
- Electron stack

**Work flow**
1. Is preset # of histories reached and both stacks are empty?
   - No
   - Yes

2. Are there enough particles in one of the stacks?
   - No
   - Yes

3. Is the preset # of histories reached?
   - No
   - Yes

4. Load particles from a stack
   - No
   - Yes

5. Generate source particles
   - No
   - Yes

6. Simulate current particles

- Start
  - Load data and transfer to GPU
  - Clear counters and stacks
    - Enough batches?
      - No
        - Perform simulation of a batch
        - Perform statistics over batches
        - Transfer data to CPU and output
      - Yes
  - End
gDPM v2.0

- Improve random number generator efficiency
  - Use CURAND, a light-weight RND generator provided by NVIDIA

- Interpolation of cross section data
  - Linear interpolation is used in gDPM v2.0
    - No loss of accuracy is observed
    - GPU support hardware interpolation

- Optimize GPU memory access
  - Use shared memory
Other Components

- Load DICOM RT format to define patient anatomy (voxel materials and structure information)
- Enable gantry, couch, collimator rotations
- Flexible source function
  - User can supplement with their own realistic Linac source model or phase space file
- Enable simulating fluence map and MLC

Dose calculation in realistic IMRT & VMAT treatment plans
Fluence map

- Fluence map: a set of beamlets $I$ with associated weights $p_I$
- Metropolis sampling

Start with an initial beamlet $I_0$

for $i = 1 \ldots N$

- Generate a trial beamlet $J$
- Generate a random number $r$
  - if $r < p(J)/p(I_{i-1})$
    - $I_i = J$
  - else
    - $I_i = I_{i-1}$
endif

Sample a particle inside the beamlet $I_i$ uniformly

end
Electron Cases

- Electron point source, 20 MeV
- Results
**Photon Cases**

- Photon point source, 6 MV spectrum
- Results

![Diagram of Photon Cases]

**Results**

- **Photon point source, 6 MV spectrum**

![Graphs showing D (MeV/g) vs. z (cm) and x (cm) for different depths]**
RapidArc Cases

- Photon point source, 6 MV spectrum
- 2 arcs
**IMRT Case**

- Photon point source, 6 MV spectrum
- 8 non-coplanar beams

Uncertainty is amplified by 50 times for clear visualization
## Results

Average relative uncertainty $\langle \sigma_D / D \rangle$ (computed in region where $D > 0.5D_{\text{max}}$),

Passing rate $P_t$.

<table>
<thead>
<tr>
<th>Source type</th>
<th># of Histories</th>
<th>Case</th>
<th>$\langle \sigma_D / D \rangle$ CPU (%)</th>
<th>$\langle \sigma_D / D \rangle$ GPU (%)</th>
<th>$P_t$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20MeV Electron</td>
<td>$2.5 \times 10^6$</td>
<td>water-lung-water</td>
<td>0.99</td>
<td>0.98</td>
<td>99.9</td>
</tr>
<tr>
<td>20MeV Electron</td>
<td>$2.5 \times 10^6$</td>
<td>water-bone-water</td>
<td>0.98</td>
<td>0.99</td>
<td>100.0</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>water-lung-water</td>
<td>0.71</td>
<td>0.72</td>
<td>98.5</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>water-bone-water</td>
<td>0.64</td>
<td>0.64</td>
<td>96.9</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>VMAT HN patient</td>
<td>N/A</td>
<td>0.88</td>
<td>N/A</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>VMAT Prostate patient</td>
<td>N/A</td>
<td>0.78</td>
<td>N/A</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>IMRT HN patient</td>
<td>N/A</td>
<td>0.57</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**CPU:** Intel Xeon processor with 2.27GHz  
**GPU:** NVIDIA Tesla C2050
Results

Execution time $T$, and speed-up factor $T_{CPU}/T_{GPU}$ for four different testing cases.

<table>
<thead>
<tr>
<th>Source type</th>
<th># of Histories</th>
<th>Case</th>
<th>$T_{CPU}$ (sec)</th>
<th>$T_{GPU}$ (sec)</th>
<th>$T_{CPU}/T_{GPU}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20MeV Electron</td>
<td>$2.5 \times 10^6$</td>
<td>water-lung-water</td>
<td>117.5</td>
<td>2.05</td>
<td>57.3</td>
</tr>
<tr>
<td>20MeV Electron</td>
<td>$2.5 \times 10^6$</td>
<td>water-bone-water</td>
<td>127.0</td>
<td>1.97</td>
<td>64.5</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>water-lung-water</td>
<td>1403.7</td>
<td>18.6</td>
<td>75.5</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>water-bone-water</td>
<td>1741.0</td>
<td>24.2</td>
<td>71.9</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>VMAT HN patient</td>
<td>N/A</td>
<td>36.5</td>
<td>N/A</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>VMAT Prostate patient</td>
<td>N/A</td>
<td>46.7</td>
<td>N/A</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$2.5 \times 10^8$</td>
<td>IMRT HN patient</td>
<td>N/A</td>
<td>48.0</td>
<td>N/A</td>
</tr>
</tbody>
</table>

CPU: Intel Xeon processor with 2.27GHz
GPU: NVIDIA Tesla C2050
## Results

- Multi-GPU implementation
  - Bash script to submit job to 4 GPUs (2 GTX590)
  - Summation and statistics are performed

<table>
<thead>
<tr>
<th>Source type</th>
<th># of Histories</th>
<th>Case</th>
<th>$T_{\text{GPU}}$ (sec)</th>
<th>$T_{\text{4GPU}}$ (sec)</th>
<th>$T_{\text{GPU}}/T_{\text{4GPU}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6MV Photon</td>
<td>$4 \times 10^9$</td>
<td>water-lung-water</td>
<td>312.82</td>
<td>78.4</td>
<td>3.99</td>
</tr>
<tr>
<td>6MV Photon</td>
<td>$4 \times 10^9$</td>
<td>water-bone-water</td>
<td>403.75</td>
<td>101.19</td>
<td>3.99</td>
</tr>
</tbody>
</table>
gCTD/gMCDRR project

- Fast kV MC simulation for CT/CBCT scans
  - gCTD: assess radiation dose received during CT scans
  - gMCDRR: simulate x-ray projections
- Developed based on gDPM but with simpler physics
  - Only photon transport
  - Secondary particle is not needed, so no stack
- gCTD: record dose to voxel
- gMCDRR: record photon energy fluence at imager
Source simulation

- Generate source particle energy according to a known spectrum
  - Generate particles in the entire energy range
  - Simulate each energy bin sequentially

- Generate source particle direction according to measured air scan
  - Using Metropolis algorithm
  - Example: full fan bowtie filter

(measured vs. simulated images)
gCTD Results

- NCAT phantom
  - CBCT scan, gantry angle: 0~360 degree

Uncertainty is amplified by 50 times for clear visualization
gCTD Results

- Head-and-Neck patient
  - CBCT scan, gantry angle: 0~200 degree

Uncertainty is amplified by 10 times for clear visualization.
gCTD Results

<table>
<thead>
<tr>
<th>Case</th>
<th>Resolution</th>
<th>$\langle \sigma_{D/D} \rangle$ (%)</th>
<th>T(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCAT</td>
<td>128×128×60</td>
<td>0.47</td>
<td>57.5</td>
</tr>
<tr>
<td>HN Patient</td>
<td>256×256×160</td>
<td>0.67</td>
<td>128</td>
</tr>
</tbody>
</table>

- Uncertainty is computed in high dose region, $D > 0.3D_{\text{max}}$
- $10^9$ particles simulated using NVIDIA C2050
gMCDRR Results

- Head-and-Neck patient

- $10^{10}$ particles simulated, $\sim 10$ min on Tesla C2050
gMCDRR Results

- NCAT phantom

- $10^{10}$ particles simulated, ~10 min on Tesla C2050
Conclusion

- **gDPM**: dose calculation for a realistic plan within 1 min or less (with multi-GPU)
- **gCTD/gMCDRR**: fast dose calculation and kV image simulation for CBCT
- **GPU is powerful for MC simulation in radiotherapy**
  - **Pros**: inexpensive, very powerful
  - **Cons**: Requires careful implementation
- **Cons**: Not as straightforward as using MPI on a cluster
- **Cons**: Rewriting/restructuring code is sometimes needed
Acknowledgement

• Supervisor: Steve Jiang
• Carrie Jiang, Mike Folkerts
• All group members
• SDSC: Dongju Choi, Amit Majumdar