Project: Assessing Medical Imaging Algorithms For GPU Systems

By Wilson Tang

Project Abstract:

Medical Imaging Applications can be assessed for GPU processing suitability through 5 main criteria: data parallelism, thread count, branch divergence, data memory required, and data synchronization. Furthermore the time-sensitive nature of the application can be further used to assess the priority in which work should be focused on adapting and optimizing these algorithms to GPU systems.

Project Goals:

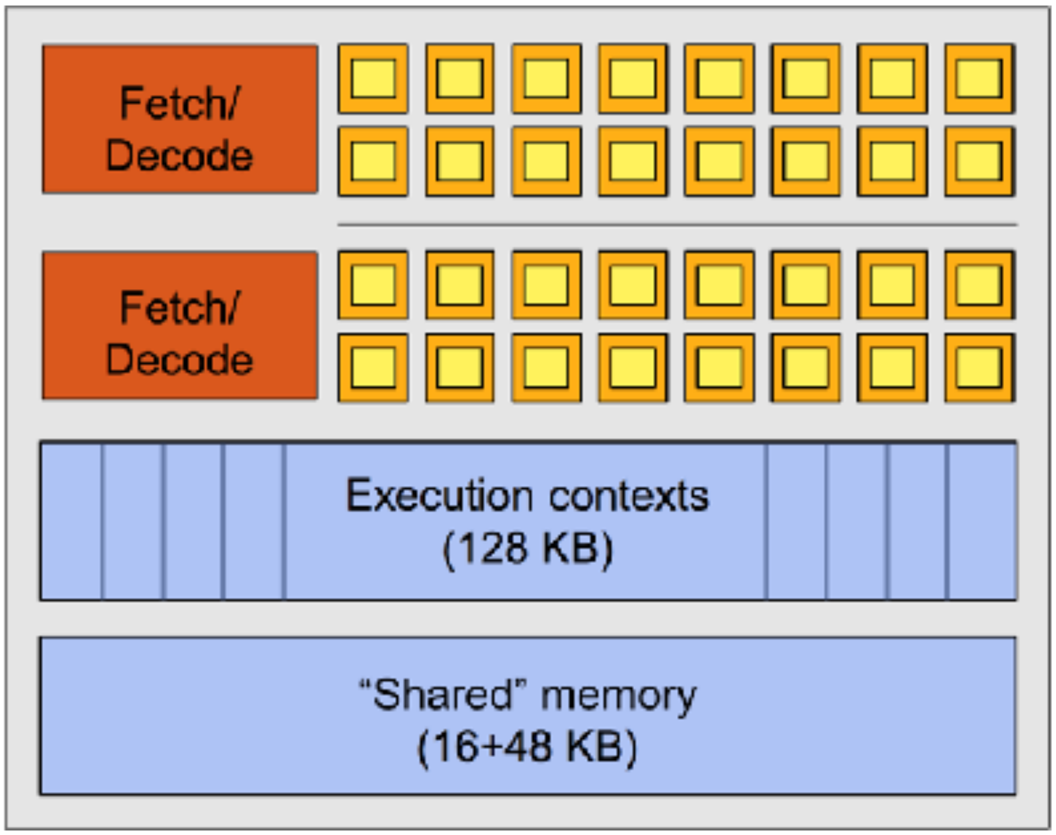
To understand basic GPU architecture and concepts as well as suitability of different medical imaging tasks for GPU processing through analyzing different algorithms.

Introduction to GPUs:

Graphics Processing Units are more specialized processing units that are designed and typically utilized for graphic intensive tasks. Today they are a highly parallel programmable feed-forward pipeline processor ideal for applications requiring large computational requirements, multiple task and data parallelism, and a focus on throughput over latency. Graphical processing applications are suited for GPUs due to the billions of simultaneous operations required to render pixels in real time.

GPU Architecture:

A GPU pipeline is typically input a list of geometric primitives, such as triangles in a 3D coordinate system. Through multiple steps the primitives are shaded and mapped onto a screen, and finally assembled into a picture. For a more in-depth explanation of GPUs see [2].



Modern GPUs typically have multiple smaller SIMD function units (small orange boxes), fetch/decode instruction blocks, a execution context memory, and a shared memory. (see figure below)

Figure 1: NVIDIA GeForce GTX 580 Core

What Makes an Medical Algorithm Good for GPUs:

An obvious usage of GPUs is in image processing applications in the medical space which have large data sets with complex computational requirements.

There are five criteria, which can represent the most important factors affecting GPU performance: data parallelism, thread count, branch divergence, data memory, memory synchronization.

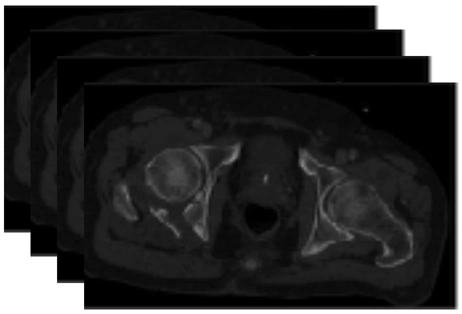
1. The highly data parallel architecture of the GPU makes it ideal for algorithms that perform the same instruction on multiple data elements in parallel, and can execute different instructions in parallel.

2. The thread count, how many individual parts a calculation can be divided into and performed in parallel should be high in order to obtain a substantial speedup of a data parallel algorithm on the GPU. A high number of threads ensures that some threads are ready while the other threads wait to deal with latency restrictions.

3. Branch divergence cause problems for a GPU because the branch command requires all threads with the same control unit to perform the same command. In general branches will reduce the performance of a GPU.

4. Memory dedicated to data on GPUs is limited and may be insufficient for algorithms that operate on large datasets or iterative methods which would require the streaming back and forth of data.

5. Data synchronization is a step required in most parallel algorithms. This can be done through atomic operations in GPUs, which forces other threads to wait for the atomic operation to execute first (resulting in serial not parallel execution).



In addition, we can add a sixth category: the time-type constraints of the type of work the algorithm is designed to do, to assess the usefulness of GPU implementations.

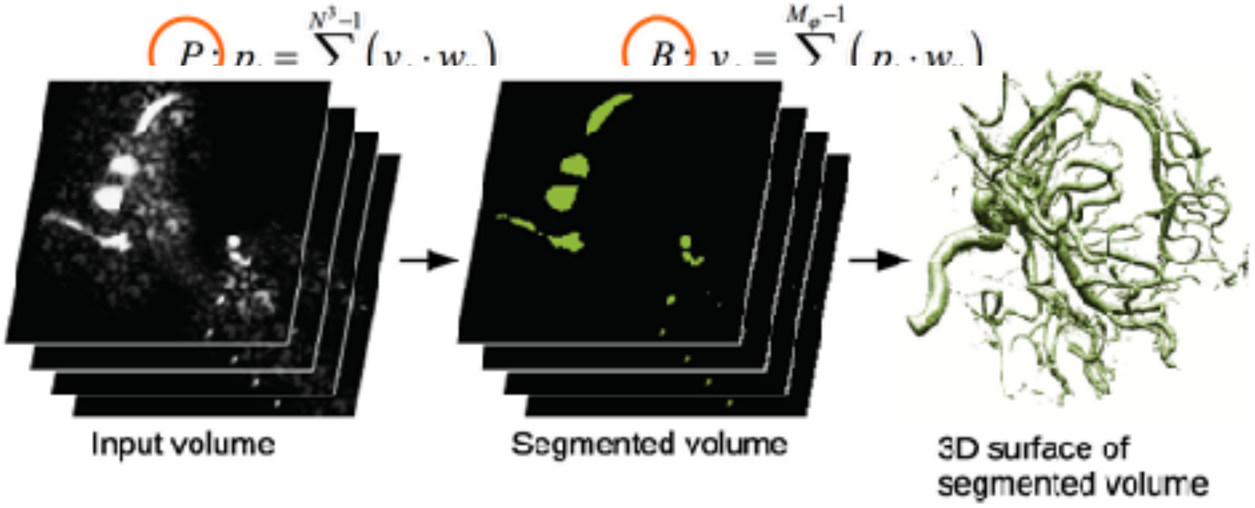
6. Different medical imaging requires processing on different scales, some need to be real-time in order to be as effective while others can be analyzed later.

Generally, an algorithm should be data parallel, have many threads, no divergent branches, use less memory than the total amount of memory on the GPU and use as little synchronization as possible. We can combine these five categories with the sixth to look at what sort of algorithms should be prioritized for implementation and optimization in the clinical world today.

Example 1: Reconstruction of Computed Tomography Scans

Computed Tomography Scans (CT Scans) are one of the oldest and most widely used medical imaging technology. The procedure takes a series of x-rays of a region of interest from multiple different angles by rotating around the body. These images are used to assemble an image of a cross-section of the body, which can be combined to create a 3D image of the a section of the body. (see diagram below). These images have pixels colored in terms of relative radiodensity, which allows for identification of different types of tissue. When the 2-D pixel is combined with the CT slice thickness a 3D unit called a voxel is presented.

To reconstruct these images from a finite number of x-ray projections the detectors must measure a line integral of the X-ray attenuation through the scanned object. An image is generated by using back projection or an inverse Radon transform to reconstruct the original density of the image. Filtered projections are mapped to voxels(v_j) in parallel and used to reconstruct a 3D image.



Assessing the Filtered Back-Projection Algorithm:

1. Data Parallelism: High, the FBP algorithm can be executed on multiple scanner projections simultaneously before aggregating the data.

2. Thread Count: High, Equal to the number of scanner projections needed to be processed.

3. Branch Divergence: None

4. Data Memory: High. Storage needed for multiple steps and the final image.

5. Data Synchronization: Low, Data will need to be synchronized after the initial processing.

6. Medical Image Timing: Urgent, CT scans can be used in diagnosing injuries such as internal bleeding and complicated fractures after accidents requiring low turn around times.

Example 2: Medical Image Segmentation

Medical Image Segmentation is an important image processing technique to divide individual elements of an image or volume into groups with common properties. (e.g identifying an organ or structure)

Thresholding segments each voxel in the slice by assessing the voxel intensity with a threshold. The simplest binary segmentation follows the following form:

$$S(\vec{x}) = \begin{cases} 1 & \text{if } I(\vec{x}) \geqslant T \\ 0 & \text{else} \end{cases}$$

Where T is the threshold, I(x) is the intensity, and S(x) is the resulting labeling.

Assessing the Thresholding Algorithm:

1. Data Parallelism: High. Completely data parallel, each voxel can be classified independently.

2. Thread Count: High. Equal to the number of voxels/pixels processed.

3. Branch Divergence: None

4. Data Memory: Low. Only data needed to be stored is the S(x), labeled results

5. Data Synchronization: None. No need for synchronization due to complete parallel segmentation

6. Medical Image Timing: Urgent to Medium. Segmentation is typically used as a further diagnosis step in assessing serious diseases.

Conclusion:

From the analysis done above, the Thresholding Algorithm used for image segmentation is extremely suitable for GPU processing due to its high data parallelism, high thread count, no branch divergence, small data memory, and no data synchronization. The Filtered Back Propagation is suitable for GPU processing due to its high data parallelism, high thread count, no branch divergence but will require a fairly large data memory and some synchronization compared to the Thresholding Algorithm. However, the Filtered Back Propagation method would have larger clinical impact due to the more time-sensitive nature and should likely be the priority. Clinical implementation will likely be especially restricted by space due to most medical data systems housed on site.

Project Reflection and Experience:

The project was designed to learn more about specialized computer architecture process design and utilization decisions. I utilized the first half of the project to learn the basics of GPU architecture before narrowing the project to focus on learning on how to assess the usefulness of using GPU to process different medical algorithms. The second half of the project was spend looking into multiple papers on the GPU computing and medical imaging processing. I adjusted a general framework for analyzing the suitability of image processing algorithms and used this to assess two common algorithm used in a CT scan reconstruction: Back Projection Algorithm and Thresholding Algorithm. By analyzing algorithm suitability for GPUs it let me gain a deeper understanding into the key architecture limitations and strengths of GPU systems. In addition, I did a basic run through of thinking how to do a CUDA implementation in order to understand assess the algorithm using the framework.

Other Applications and Future Work:

Potential extensions for this project that I envision would be to take the assessment of algorithms a step further by attempting to implement them in a modern GPU. This would be a great way to learn about general GPU programming such as CUDA as well as investigate the exact effectiveness of running these algorithms on a GPU based system. From my readings these implementations could have 1-2 orders of magnitude difference in time while maintaining a similar quality, which could make a huge difference in the clinical setting.

Bibliography:

[1] E. Smistad, T.L. Falch, M. Bozorgi, A. Elster, F Lindseth
Medical Image segmentation on GPUs - A comprehensive review
Med. Phys., 20 (2015), p.1-18,

[2] G. Pratx, L. Xing
GPU computing in medical physics: a review
Med. Phys., 38 (2011), p. 2685,

[3] J. Owens, M. Houston, D. Luebke, S. Green, J.Stone, J. Phillips
GPU computing
Proc. IEEE, 96 (2008), pp. 879-899,

[4] K. Mueller, F. Xu
Why Do GPUs Work So Well Work So Well for Acceleration of CT? for Acceleration of CT? SPIE Electronic Imaging '07 Keynote, Computational Imaging V. *Presentation*