

Sean Kirby, University of Maryland, Eastern Shore
William Kostan, University of Virginia
Dr. Arthur J. Lembo, Jr., Salisbury University

Introduction:

A continuing challenge in GIScience and many other fields is the efficient and economical processing of massive data sets. We describe how to leverage massive parallel processing through general purpose computing using the graphics processing unit (GPGPU) for raster GIS operations. Using video-gaming cards to create parallel processing solutions within the GPU is relatively new and almost nonexistent in GIScience. By designing a software add-on built to run select raster analysis operations on a CUDA-enabled GPU, we were able to achieve a 7x speed improvement compared to traditional (ArcGIS) raster processes. In addition, our approach allows us to perform raster processing in under an hour for 93GB files with over 25 billion pixels. Finally, we identify key areas of future work to improve execution time.

Data and Methods:

Fourteen ESRI FLT raster files (ranging from 24MB to 12GB) were processed and analyzed using 3x3 kernel functions for slope, aspect, terrain ruggedness, min, max, range, and mean. These functions were chosen as they represent “embarrassingly” parallel functions that can be run independent from one another, ideal for GPU processing. Each file was processed 20 times to determine the average speed using both ArcGIS and CUDA-C. The CUDA-C process (Figure 1) included:

1. Partition data by rows and read into an array in memory*
2. Copy data array to GPU.
3. Launch GPU kernel for raster processing.
4. Process data in parallel in GPU through a combination of blocks and threads.
5. Upon completion of each kernel, copy data from GPU to the CPU.
6. Write output to the appropriate file (one file per raster function)
7. Additional kernels are launched from data already on the GPU

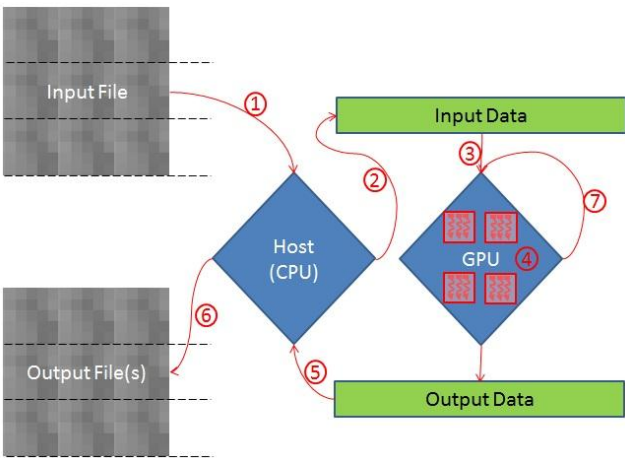


Figure 1: CUDA execution model

* Additional speed improvements may be achieved by running this step as separate, parallel threads on the CPU, thereby allowing the CPU to prepare the next data block while the GPU is processing data

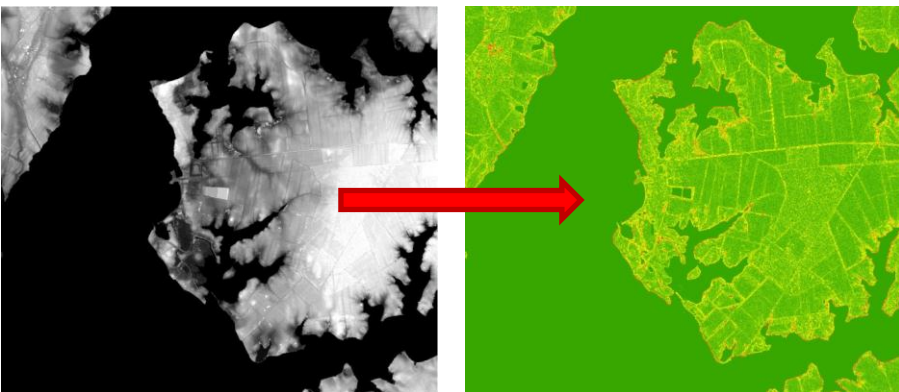


Figure 2: Oxford County, MD, LiDAR Imagery of Oxford County before (left) and after (right) slope transformation.

Results and Discussion:

On average, the introduction of CUDA provided a 7x improvement over the same function used in ArcGIS.

Execution time for the CUDA functions exhibited a very stable and predictable linear trend for all functions over the various size raster files, allowing us to better predict response times for larger data sets. To test the linear fit we forecasted execution time out to a 93GB (25 billion pixels) raster and predicted an execution time of 3100. Our actual processing of the file completed in under 3000 seconds. Unfortunately, the larger file was too large to process in ArcGIS so a comparison between the products is impossible to determine.

In contrast, while most of the execution times for ArcGIS appear linear with respect to size, both slope and aspect execution deviate substantially from a linear fit when the size of the input raster was between 1.4GB and 4.0GB. We suspect that the deviation is related to the ESRI ADF format’s separation of data into files between 1.2 GB and 2GB or separate algorithm for datasets over some threshold size.

The greatest bottleneck was obtaining data from the disk and moving to RAM (Figure 1, step 1). Once data was on the GPU, parallelization took over and made short work of the processing. Numerous strategies were used to improve the data acquisition step with noticeable results. Testing under other hardware configurations confirmed that having more GPU memory to load larger chunks of data played a bigger role in execution than having more processing streams.

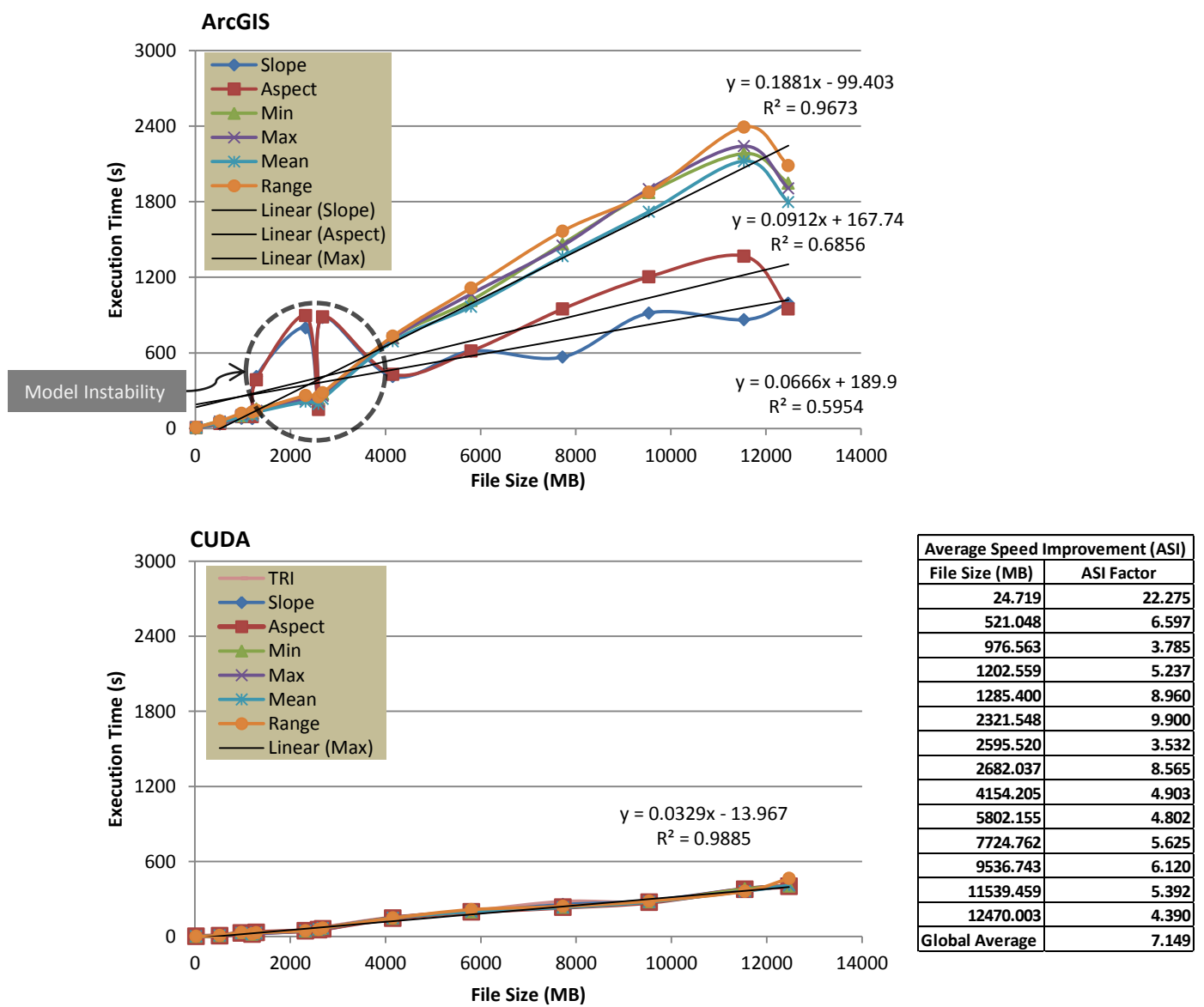


Figure 3: Comparison of ArcGIS and CUDA-C generated performance of several raster functions as measured by execution time show that CUDA processes the data 3-22 times faster than the corresponding ArcGIS process. In addition, the CUDA process was perfectly linear ($R^2 = .98$), while the ArcGIS process exhibited serious instability between 1.25GB and 4GB for Slope and Aspect.

Final Software Package:

One of the primary goals of the project was to create a simplistic add-on to ArcGIS for dramatic performance improvement. This was accomplished by incorporating the CUDA executable into a model in ArcToolbox for use any computer with ArcGIS and a CUDA-enabled GPU

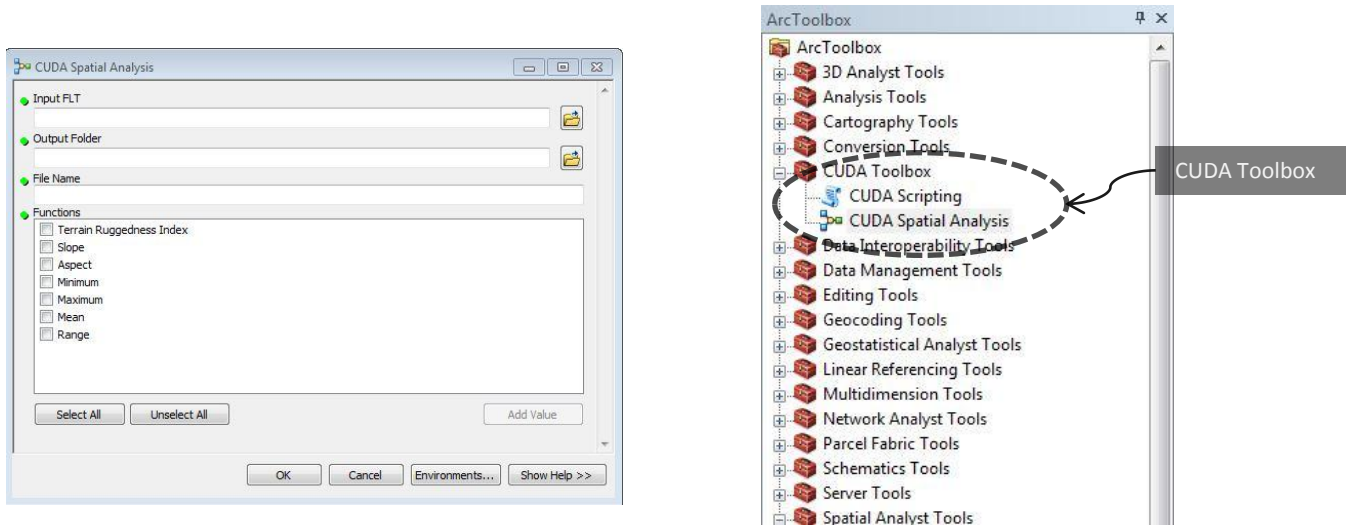


Figure 4: CUDA program integrated into ArcMap 10.0 as a toolbox with full graphical user interface (GUI)

Conclusions and Future Considerations:

Parallel processing of raster functions were 3-22 times faster than ArcGIS depending on file size. Also, processing multiple functions on a single dataset achieved even greater performance gains (Figure 4). A final test included slope processing of a 93GB, 25 billion pixel raster file in under 3000 seconds, indicating continued linear performance even with enormous file sizes.

Future work should include utilization of parallel CPU threads for reading data into RAM while the GPU processes data, which we believe will provide the most dramatic performance increase. Additionally, because the greatest efficiencies were demonstrated when performing multiple computations for the data on the GPU, future work should include exploration of similar models in climatology, ecology, remote sensing, and natural resource assessment. There are many GIS models that require multiple iterations that can see enormous speed improvements if redesigned within a GPU context.

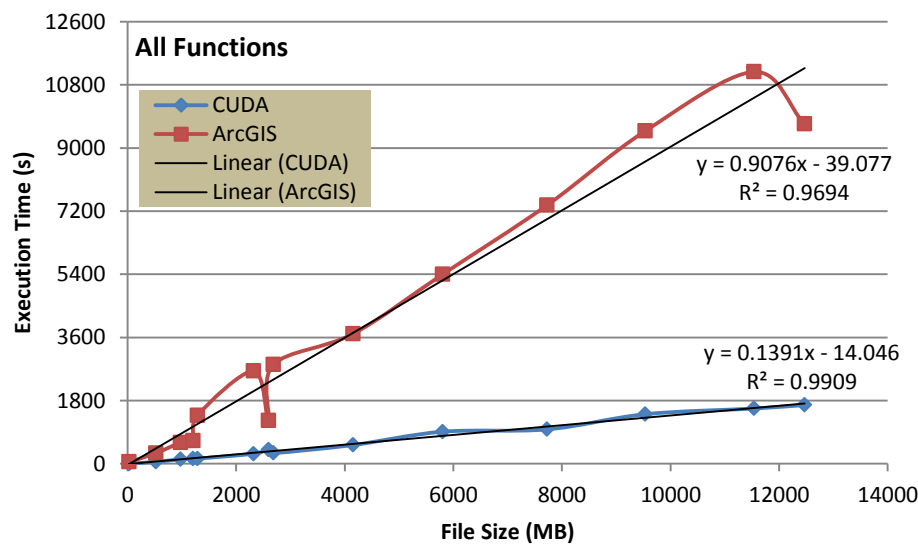


Figure 5: CUDA vs. ArcGIS performance in executing six spatial analysis transformations per dataset as measured by execution time

Acknowledgments:

Research was funded by the National Science Foundation under grant number CCF-1156509 and hosted by Salisbury University.