Sensitivity Analysis of GP-GPU Technology in its Application to GIS Terrain Analysis

David Eberius, Salisbury University
Dr. Arthur J. Lembo, Jr., Salisbury University

Introduction:

Research by Knipprath and Lembo (2013) showed that while CUDA methods for terrain modeling are faster than serial methods, only moderate gains were seen when attempting to further improve CUDA processing through multi-threading, novel tiling schemes, and memory mapped files. The primary limitation for improving speed appears bound to the I/O bottleneck when compared to the fast speed of the GPU computation. Traditional terrain modeling is an I/O heavy endeavor, with small numbers of computations for a massive number of pixels, thus exceeding the memory limitations of the GPU.

This work sought to investigate speed improvements when increasing the number of computations per pixel, as opposed to increasing the number of pixels themselves. To accomplish this, slope analysis was performed using 3x3, 5x5, 7x7, and 9x9 kernels while keeping the number of computations the same.

The results showed that having smaller datasets with larger numbers of computations per element are better suited for CUDA based GIS analysis.

Data and Methods:

This work was specifically interested in testing the speed difference when using a CUDA kernel with many more computations. To achieve this, the number of raw computations remained constant (within .03%) between the kernel configurations by scaling the data size (i.e. A 9x9 kernel has ~9.9 times more computations per pixel than a 3x3 kernel, so the amount of data for the 3x3 kernel is ~9.9 times larger than the 9x9 kernel resulting in equivalent numbers of computations).

The starting data size was for the 9x9 kernel (the kernel with the smallest data size because it has the most computations) was ~5.637 Mb. Thus, the other kernels were scaled accordingly such that the 3x3 kernel ended up being ~55.81 Mb. Each kernel configuration was run 10 times for this file size and the average computed. The initial file size was then increased by a factor of two, three, etc… until a factor of 8. The various kernels were then run for the new size (Table 3).

Results and Discussion:

As shown in Table 1, the 3x3 CUDA kernel implementation was ~83-100 times faster than the CPU implementation depending on the amount of data. As the number of computations is increased with the same data size by increasing the kernel size, the speed increase of the GPU goes down. However, the GPU is still much faster than the CPU with the lowest increase being ~48 times. This decrease is most likely due to the fact that the CPU is much faster per calculation due to its much higher clock rate.

Each subsequent increase in kernel size does not increase the number of computations in a linear fashion. Similarly, the trend of decreased run time as the number of computations increases is not linear with 5x5 ~33% faster than 3x3, 7x7 ~17% faster than 5x5, and 9x9 ~2.5% faster than 7x7 on average. The 9x9 implementation even ran slower than the 7x7 implementation on some occasions.

Conclusions and Future Considerations:

When comparing the CPU and GPU implementations, it becomes clear that for a parallelizable problem, particularly an embarrassingly parallel problem such as this, a GPU implementation runs much faster.

As shown in Table 3, increasing the number of computations per kernel will complete more computations per second than having fewer computations per kernel up to a certain point. We believe this is due to the overhead of spawning new threads, which becomes a bottleneck when performing few calculations per kernel. However, increasing the number of computations per kernel can eventually become a bottleneck as shown by the steady lessening of performance increase as computations are increased (Table 3), as evidenced by 9x9 implementation being slower than 7x7 implementation in some instances.

I/O is the largest overall bottleneck. Having more computations per kernel can increase the number of computations per second. Future work in this area may focus on keeping data in memory and doing more computations per kernel for optimized GPU usage.

3x3 Kernel

1	2	1
2	0	2
1	2	1

5x5 Kernel

1	1	2	1	1
1	3	4	3	1
2	4	0	4	2
1	3	4	3	1
1	1	2	1	1

7x7 Kernel

1	1	1	2	1	1	1
1	3	3	4	3	3	1
1	3	5	6	5	3	1
2	4	6	0	6	4	2
1	3	5	6	5	3	1
1	3	3	4	3	3	1
1	1	1	2	1	1	1

9x9 Kernel

1	1	1	1	2	1	1	1	1
1	3	3	3	4	3	3	3	1
1	3	5	5	6	5	5	3	1
1	3	5	7	8	7	5	3	1
2	4	6	8	0	8	6	4	2
1	3	5	7	8	7	5	3	1
1	3	5	5	6	5	5	3	1
1	3	3	3	4	3	3	3	1
1	1	1	1	2	1	1	1	1

Figure 1. The different kernels used for slope analysis have varying degrees of weights with the 9x9 kernel having considerably more complexity than 3x3.

Kernel Size	Data Size (Mb)	CPU Time (ms)	GPU Time (ms)	Speed-Up Factor
3x3	5.637	286.47	3.46	82.80
3x3	140.925	7253.81	72.37	100.81
5x5	5.637	407.39	6.11	66.65
5x5	140.925	10117.69	126.72	80.15
7x7	5.637	568.86	10.42	54.61
7x7	140.925	14347.46	216.44	66.45
9x9	5.637	813.24	16.69	48.73
9x9	140.925	20545.62	345.18	59.60

Table 1. Results of speed improvements when more computations are performed on the GPU over the CPU.

Source	3x3 Calculations	5x5 Calculations	7x7 Calculations	9x9 Calculations
Logical Comparisons	15	31	55	87
Calculations in Indices	40	130	300	572
Inverse Tangent	1	1	1	1
Square Root	1	1	1	1
Exponent	2	2	2	2
+ - / *	18	45	85	135
Assignment	4	4	4	4
Total	81	214	448	802

Table 2. A breakdown of the number of calculations for 3x3, 5x5, 7x7, and 9x9 Kernels broken down by source. Clearly the complexity increases rapidly as you increase kernel size.

File Size Multiplier	9x9 vs 7x7	7x7 vs 5x5	5x5 vs 3x3	7x7 vs 3x3	9x9 vs 3x3
1	-0.38%	9.61%	32.29%	38.80%	38.57%
2	-5.47%	17.26%	36.97%	47.85%	45.00%
3	2.58%	17.25%	32.20%	43.90%	45.35%
4	1.52%	17.76%	31.75%	43.87%	44.73%
5	3.61%	20.32%	29.53%	43.85%	45.87%
6	5.64%	17.74%	33.48%	45.28%	48.37%
7	6.80%	17.75%	33.30%	45.14%	48.87%
8	5.98%	17.61%	32.98%	44.78%	48.08%
Average	**2.54%**	**16.91%**	**32.81%**	**44.18%**	**45.60%**

Table 3. This shows the percent increase or decrease seen when comparing a kernel configuration with more computations to one with fewer.