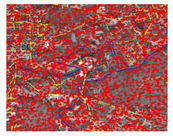
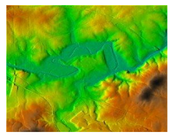
A General Purpose Graphics Processing Solution to Quadtree Creation

Joshua Gluck, advisor: Andrew Danner; Swarthmore College, PA

Overview

- Get hi-resolution spatial data from Light Detection and Ranging (LIDAR) technology
- Leverage high computational throughout of General Purpose Graphics Processing Units(GPGPUs) and minimize overhead costs with CUDA for rapid computing of a Quadtree from LIDAR data
- Codify the optimizations used in GPGPU Quadtree creation for other problems



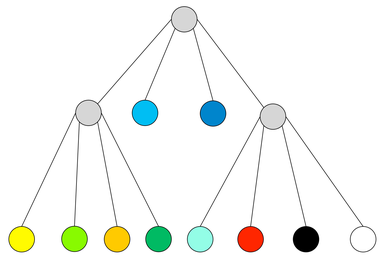


What is a Quadtree?

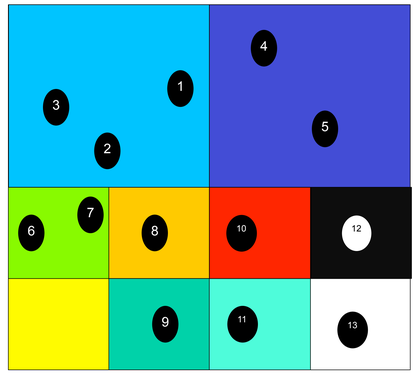
- A Quadtree is a data structure used to spatially index points in R^2
- No Quadtree node can contain more than a user defined K points. Nodes with more than K points divide into 4 sub-nodes, and the points are stored in the sub-node in which they would be located
- Searching a Quadtree for a given point is considerably more efficient than searching an unsorted array of points.

Our Implementation

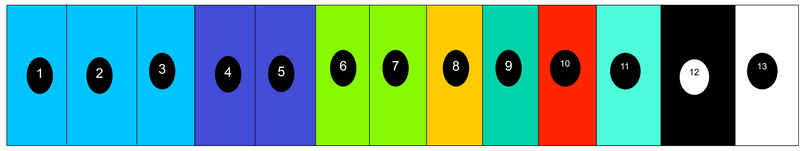
- An array based Quadtree
  - Each node consists of a start indices and length
  - Points within a node are stored in its segment of the array
- Does not require copying the point data from CPU to GPU more than once
- Allows hundreds of threads to execute in parallel, while avoiding conflicts over shared resources
- No limitations on the depth of the tree



Tree Representation of a Quadtree. Gray nodes are non-leaf nodes



Geographic Quadtree representation



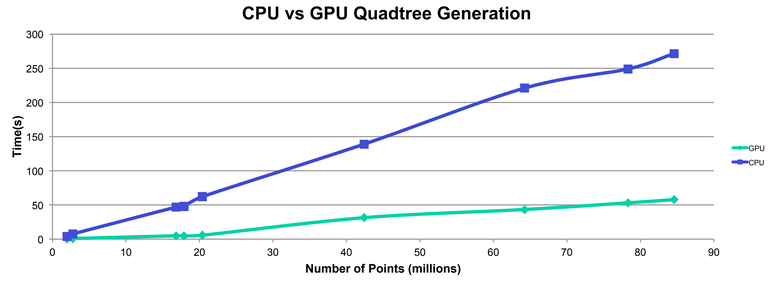
The array implementation of the above Quadtree

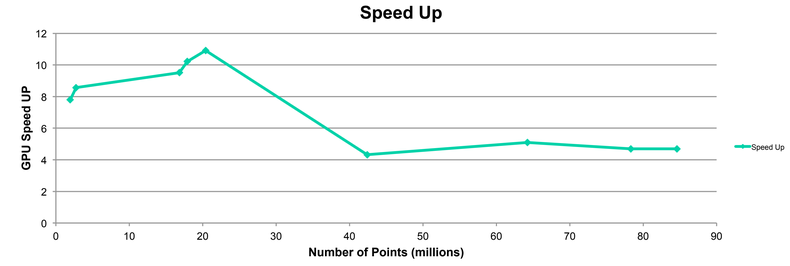
Our Work

- Utilize both block and thread level parallelism on the GPU for maximum speed gains, toggle between thread and block implementations based on problem characteristics
- Generate specific read indices for each thread based on its thread ID and block number to prevent read overlaps
- Generate unique offsets for each thread in a block based on the offset of the thread just before it to prevent write overlaps
- Based on general statistics, remove child nodes which do not need further processing, using RADIX sort

Test/Results

- Tested GPGPU implementation on a Nvidia Quadro 1000M GPGPU against single core CPU approach
- Tested on data sets ranging from 2 million to 85 million LIDAR points





Analysis

- The results above show the strong gains, 4x to 10x, that an efficiently implemented GPGPU system can have over a CPU based approach
- Due to GPU memory limits, larger experiments require more CPU ⇔ GPU communication, limiting speed up.

Conclusions

- The computation time gained from using the GPGPU solution here showcases the overarching utility of GPGPU solutions to non-graphics and non-embarrassingly parallel problems.
- Future Work will include the expansion of this system into a distributed framework for even greater computational gains, as well as techniques to reduce the effects of GPU memory constraints

References

- Zhang, Jianting, Simin You, and Le Gruenwald. "Indexing large-scale raster geospatial data using massively parallel GPGPU computing." *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2010.
- M. Kelly, A. Breslow, Quad-tree Construction on the GPU: A Hybrid CPU-GPU Approach