General neighborhood analysis model for grid DEM on CUDA

Yong Gao*, Hao Yu, Lei Liu, Xiao Guo

Institute of RS & GIS, Peking University, Beijing, China
Telephone: 86-010-62751186
Email:gaoyong@pku.edu.cn

1. Introduction

Neighborhood analysis on the Grid Digital Elevation Model (DEM), such as, slope, aspect, edge detection, filter changes, etc. is a kind of basic and important spatial analysis. It calculates each target value from the nearest neighborhood grid points' value by a certain neighborhood template. Then two performance problems are brought up. First, it is computing intensive, for it requires the calculation of a neighborhood template for each grid point. Second, the DEM data size is usually big, even to GB. Therefore, high performance spatial computing technologies are required to support more effective neighborhood analysis.

GPU-based general-purpose computing (General Purpose GPU, GPGPU) is an important direction for high performance computing. The core idea is increasing the amount of computing transistors on a graphics chip, and reducing the number of caching transistors and registers. So that with the same process and the same volume of chips, GPU computing power raises a hundred times than the CPU. Nvidia put forward the Compute Unified Device Architecture (CUDA) technology, to help developers use standard C programming language to write general-purpose computing applications running on the GPU chip, without familiarities to the GPU chip instruction. However CUDA still has shortcomings because its generic computing model is not enough. GPU chip reduces the number of registers used extensively for the cache. Then the traditional CPU cache technologies, pipeline, instruction, order execution and optimization cannot be reused. Developers need to develop theri own strategy to consider how to obtain details of the data as well as the rules of the instruction execution from the memory device. Traditional CPU-based program ported to the GPU chip needs to be redesigned and recoded. The migration cost is huge. It is the main problem to make traditional algorithm applied to the CUDA environment in the area of GPU high performance research.

Currently, there are many studies to apply CUDA high-performance computing into GIS, particularly in the DEM processing, such as Fan's (2010) ground clutter fast convergence analysis of parallel algorithm simulation, convergence analysis (Zhao. 2010), IDW parallel algorithm based on CUDA and DEM (Liu E. 2011), viewshed analysis (Gao Y. 2011), Local acceleration in Distributed Geographic Information Processing(Zhao Y. 2010). However, these studies just bring CUDA into some certain fields (Ryoo S, 2008). In order to use CUDA widely in DEM, a more general solution should be proposed.

2. Modules and Methods

This paper presents a general model to process grid DEM neighborhood analysis based on CUDA. The purpose is to give a system and method for providing a common model, not only taking advantage of the GPU high-performance processing, but also reusing the same part of neighborhood analysis process on CUDA. By reusing the same part, we could simplify the development. Our model consists of four modules: the data IO module, function scheduling module, kernel function module and neighborhood analysis operator.

The data IO module exclusively occupies one thread, which is called the IO thread here, supporting data read and write operation. Function scheduling module is responsible for coordinating data IO thread and the CUDA kernel function (to be executed on CUDA). It also allocates more than one memory block used as buffer. Multiple threads are started at the same time, one is responsible for data IO, namely IO threads; the remaining threads are called worker threads. The number of worker threads is equal with the number of the GPU chips on the host machine. Each GPU chip is corresponding to each worker thread, making data IO and CUDA kernel function performs parallel execution. Kernel function module is responsible for the data migration between main host memory and the GPU chip memory. Kernel function also executes the neighborhood analysis operator on the GPU chips, after the data is copied into GPU memory. Neighborhood analysis operator is called by kernel function to execute each point's calculation, in one template. The operator is an abstract interface to developers to be fulfilled by developers. Developers for different neighborhood analysis applications only need to implement the different programming code within a neighborhood template, and the other parts of the whole process are the same and transparent to developers. So the process is much simplified.

The whole process is divided into following steps:

1) Fulfill the neighborhood analysis operator interface, and register the operator function pointer into the CUDA kernel function. This step is done by developers.

2) Start an IO thread, to read the raw DEM data from input file into the memory buffer. The memory buffer is divided into several blocks. When all the blocks are written, the process is blocked waiting for worker thread to process.

3) Production-consumer model: the IO thread acts as a producer, the worker thread to act as consumers. When the IO thread finishes filling data into a memory buffer, a worker thread, we call it T1, is wake up to consumer the data.

4) The worker thread T1 transfers data from the main host memory buffer to the global memory on GPU chips.

5) After data transferring, the worker thread T1 call the neighborhood analysis operator, which is the CUDA kernel function fulfilled in step 1, to process the calculation. The results are stored in GPU memory then.

6) The worker thread T1 migrates the results data from GPU memory to main host memory.

7) After results of the data are written back to the memory buffer, called B1, an IO thread is wake up.

8) The IO thread would write the B1 into the output file. And read the next block of data from input file into the memory block B1. When the B1 is updated by new raw data, it will wake up worker thread and go to step 3.

9) Repeat steps 2-8, until all input file processing is completed.

3. Experiments

We carried out some experiments. The machine we used has one CPU (DualCore Intel Pentium D 930, 3000 MHz) and a GPU (GeForce 9300 GE). CUDA version is 3.0. Four sets of data are chosen here: 6000*4000 pixels, 3750*2500 pixels, 2400*1600 pixels and 1200*800 pixels. Smooth filtering operation is employed here, for smooth filtering is a typical neighborhood analysis. The results show that the model could reach 7 times speed up.

Data size	CPU	GPU	speedup
6000*4000	108.969s	15.000s	7.3
3750*2500	42.328s	6.016s	7.0
2400*1600	17.359s	2.469s	7.0
1200*800	4.312s	0.640s	6.7

Table 1. speedup of 9*9 template smooth filtering.

4. Conclustion

The optimization is obvious in our experiments. The more important thing is based on the general model, we proposed, developers could easily implement neighborhood analysis, taking advantage of GPU computing power without much migration costs from CPU to GPU. In the future, we will combine CUDA with more kinds of GIS analysis.

5. Acknowledgements

This research was supported by The National High Technology Research and Development Program of China (Grant No. 2011AA120303).

6. References

Fan G, Huang Zh, Zhang X, Yang Zh, Fast Ground Clutter Simulation Based on CUDA and DEM Data, Modern Radar, Vol. 32 No. 9, Sep. 2010

Gao Y, Yu H, Liu Y, Liu Y, Liu, M, Zhao Y, "Optimization for viewshed analysis on GPU," Geoinformatics, 2011 19th International Conference on , vol., no., pp.1-5, 24-26 June 2011

Liu E, Wang Y, IDW parallel algorithm and experiment analysis, Journal of Geographical Information Science. Vol. 13, May. 2011.

Ryoo S, and Rodrigues C, et al. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming, Salt Lake City, UT, USA, ACM. 2008

Zhao X., Miao Q., Fu Zh., Su Ch., Li X., Research and realization in parallel algorithm of confluence analysis based on CUDA, Application Research of Computers. Vol. 27 No. 9, Jul. 2010.

Zhao Y, Huang Zh, Chen B, Fang Y, Yan M, Yang Zh, , "Local acceleration in Distributed Geographic Information Processing with CUDA," Geoinformatics, 2010 18th International Conference on , vol., no., pp.1-6, 18-20 June 2010