Hybrid implementation of evaluation of primary topographic parameters using GPU-accelerated clusters

Przemysław Stpiczyński, Dominik Szałkowski
Department of Computer Science, Institute of Mathematics
Maria Curie-Skłodowska University, Lublin, Poland
przem@hektor.umcs.lublin.pl, dominisz@umcs.lublin.pl

Leszek Gawrysiak, Łukasz Chabudziński
Department of Earth Sciences and Spatial Management
Maria Curie-Skłodowska University, Lublin, Poland
leszek.gawrysiak@umcs.pl, lchabudzinski@gmail.com

Abstract—**The aim of this paper is to present an efficient method for parallelizing computations of primary topographic parameters like aspect, slope, and curvature for very large data sets using clusters with GPU-accelerated nodes. We outline the implementation and discuss the results of experiments, which justify the use of such computer architecture.**

I. INTRODUCTION

Topographic parameters and indicators are the essential part of the spatial data analyses, which are used to show dependencies between various components of the natural environment. The parameters, which are obtained from the digital elevation model (DEM), are also used to describe terrain properties in a quantitative way. Parameters like slope, aspect, and curvature are called primary topographic parameters and they are directly computed from the DEM. The topographic position index, topographic wetness index, stream power index, and roughness are called secondary topographic parameters and they are derived from the primary ones. The values of the parameters represent important properties of the surface, which in turn determines the behavior of hydrologic, geomorphologic, and ecologic processes.

For the last twenty years the size of the GIS data to process have been constantly growing together with the complexity of computations needed to perform spatial analyses (Healey et al., 1997). There are some papers concerning the use of standard multiprocessor architectures for GIS related tasks (Huyaji W. et al., 2011, S. H. Han, 2009, F. Huang, 2011, X. Guan, 2009) as well as surveys of utilizing parallel capabilities of new architectures of general purpose GPUs (Graphics Processing Unit) (Osterman, 2012). However, there are no studies that take into account the hybrid nature of contemporary multiprocessor architectures consisting of various types of computing units, i.e. many-core CPUs and multicore GPUs. Today, this approach should be used to fully exploit the true power of high performance computers. In order to efficiently deal with very complex spatial analyses, it is necessary to develop new kinds of algorithms and computational techniques, which are based on an effective use of all processing devices (CPU and GPU cores) available within the nodes of computer clusters.

II. NEW HYBRID ALGORITHM

In this paper, we show how to use hybrid node clusters to compute primary topographic parameters like aspect, slope, and curvature for very large data sets, which exceed the computational resources of one node. We show how to store the data in cluster distributed memory and how to use different types of computational devices.

The general outline of the data processing scheme for large DEM files is as follows:

1. Divide the DEM file into overlapping horizontal stripes and write them into separate files.
2. Distribute the files among the nodes of the cluster.
3. Start the parallel program on the cluster to process distributed data. Each process running on a single node is responsible for distributing work among all available computational devices (CPU and GPU cores).
4. Locally, compute necessary primary topographic parameters (e.g. slope, aspect, curvature) in parallel using CPU and GPU cores.
5. Retrieve the processed data from all nodes.
6. Merge separate output files containing results into one large DEM file.

Although, we have considered the use of MPI (*Message Passing Interface*) for distributing the work among nodes of the cluster (Step 2 and 5), it can be simply implemented using a shell script spawned on nodes. We use routines provided by GDAL library for splitting and merging DEM files (Step 1 and 6).

In: Geomorphometry for Geosciences, Jasiewicz J., Zwoliński Zb., Mitasova H., Hengl T. (eds), 2015. Adam Mickiewicz University in Poznań - Institute of Geoecology and Geoinformation, International Society for Geomorphometry, Poznań

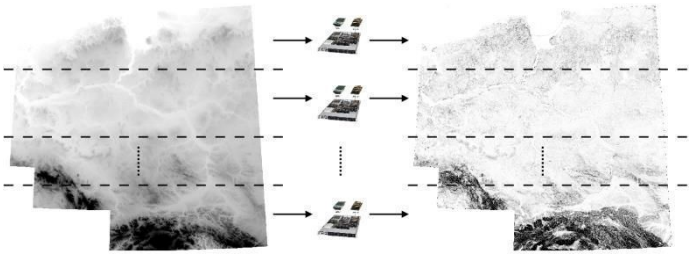

Figure 1. Data distribution scheme

Figure 2. Local computations on a single node

Data distribution scheme among available nodes is presented in Figure 1. A large input data file (presented on the left hand side) is divided into horizontal stripes (one stripe for each cluster node) with overlapping border of 1 pixel. Such a border is required due to nature of used algorithms (to compute the result for one grid cell we need its neighbours, see Figure 3). All stripes are processed in parallel. Then the results of computations are merged into one big file (right hand side of the figure).

On each node, all CPU and GPU devices are used. The data are divided into two (or three) horizontal stripes. Two stripes are always processed by two GPUs. In case of large amount of data (exceeding the size of GPU memory) the third stripe is assigned to CPU cores. This approach is presented in Figure 2.

III. DATA FILES

For our numerical experiments we use ten DEM files with properties presented in Table 1. These files contain elevation data of Poland area with various cell sizes which result in file sizes. Figure 4 shows the memory occupation of the devices used for processing the data.

TABLE 1. PROPERIES OF DEM FILES

File	Columns	Rows	Total memory (MB)	GPU0 memory (MB)	GPU1 memory (MB)	CPU memory (MB)
DTED2_31%	9954	9239	350	175	175	0
DTED2_44%	14129	13114	706	353	353	0
DTED2_54%	17340	16094	1064	532	532	0
DTED2_63%	20230	18777	1449	724	724	0
DTED2_70%	22478	20863	1788	894	894	0
DTED2_77%	24726	22949	2164	1082	1082	0
DTED2_83%	26652	24738	2515	1257	1257	0
DTED2_89%	28579	26526	2891	1445	1445	0
DTED2_94%	30185	28016	3225	1499	1499	227
DTED2_100%	32112	29805	3651	1499	1499	653

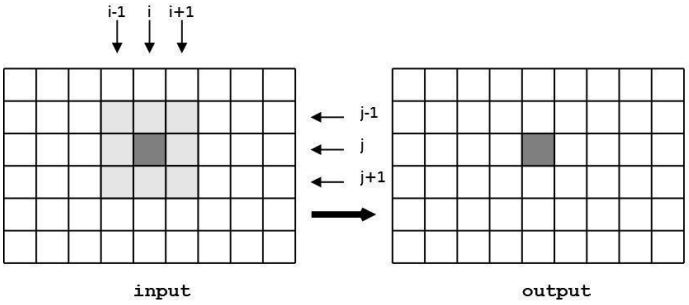

Figure 3. Using a data cell and its neighbours

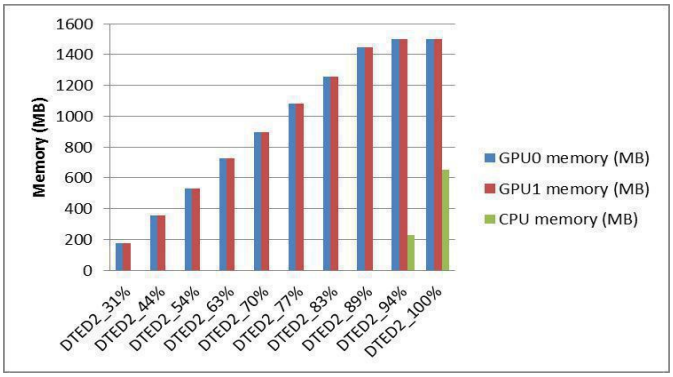

Figure 4. Memory occupation during processing

IV. SOFTWARE, HARDWARE AND RESULTS

Our software is a library of C/C++ functions for computing primary topographic parameters (slope, aspect, curvature, insolation). Inside a single node we use OpenMP and CUDA interfaces for parallel programming. This approach allows using many CPU and GPU cores at the same time, which increases the total performance of the computations. Our software also uses the widely known GDAL/OGR open source library for dealing with GEOTIFF files.

Tests were run on our Solaris cluster, which consists of 32 hybrid nodes. In each node, there are two Intel Xeon X5650 processors (6 cores each with hyper-threading, 2.67 GHz) and two NVIDIA Tesla M2050 cards (448 CUDA cores, 3 GB GDDR5 RAM with ECC off). The nodes are connected using 40 Gbit/s Infiniband. Programs were compiled using NVIDIA CUDA Toolkit version 5.5 and Intel Cluster Studio version 2013. Debian GNU/Linux operating system was used.

A comparison of the calculations performed this way with the results obtained using well known GIS programs (ArcGIS 10.1, GRASS 6.4.3, GDAL 1.10.1 and QGIS 2.0.1) on a PC with processor i7 (2.93 GHz, 4 GB RAM) shows that the algorithms are accurate and very efficient, especially in case of large data sets (see Figures 5-7).

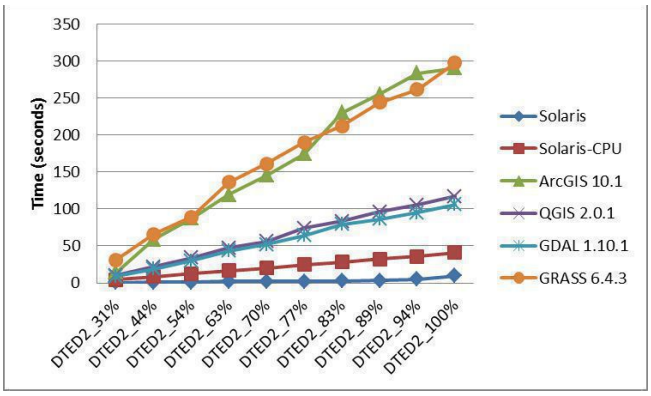

Figure 5. Execution time for slope

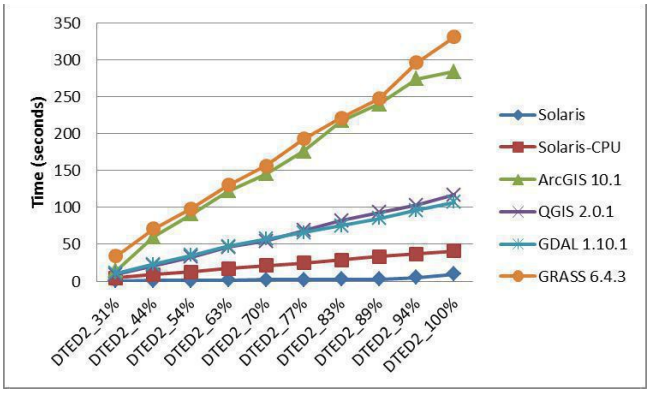

Figure 6. Execution time for aspect

Figure 8 and Table 2 present the scalability of our implementation. We can observe that the speedup grows with the increasing number of nodes. The best performance is achieved when only GPUs are used. The use of CPU cores results in the performance degradation, however it is necessary for larger data files when the amount of data to be processed by a single node exceeds the memory capacity of both GPUs.

TABLE 2. EXECUTION TIME (SECONDS) FOR VARIOUS NUMBER OF NODES

Nodes	Aspect	Aspect-CPU	Slope	Slope-CPU	Curvature	Curvature-CPU
1	9,76	41,93	9,43	40,66	5,76	17,29
2	1,83	20,36	1,88	19,98	1,81	8,74
3	1,22	13,40	1,27	13,16	1,23	5,65
4	0,94	9,92	0,93	9,59	0,89	4,23
5	0,72	7,75	0,75	7,49	0,72	3,35
6	0,54	6,02	0,62	5,76	0,59	2,64
7	0,51	4,51	0,53	4,32	0,51	1,98
8	0,44	3,58	0,46	3,47	0,45	1,58

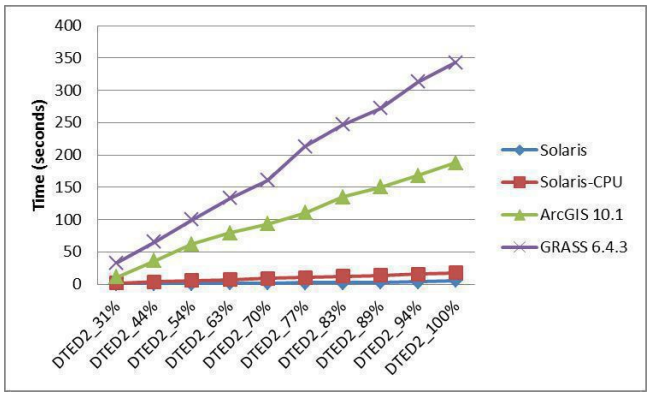

Figure 7. Execution time for curvature

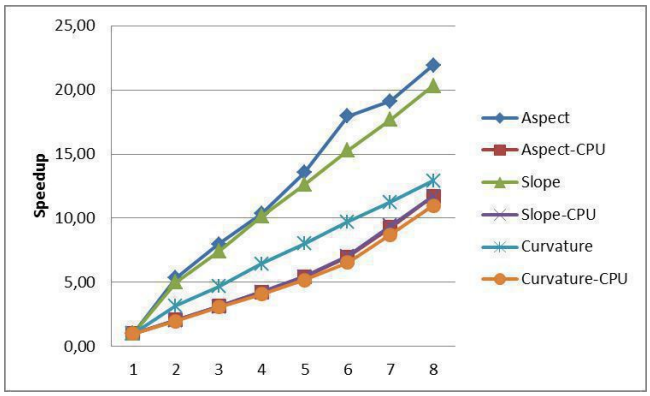

Figure 8. Speedup for various number of nodes

REFERENCES

[1] Fang Huang, Dingsheng Liu, Xiaowen Li, Lizhe Wang, Wenbo Xu: 2011: Preliminary study of a cluster-based open-source parallel GIS based on the GRASS GIS. Int. J. Digital Earth 4(5): 402-420

[2] Healey R.(ed.), Dovers S.(ed.), Gittings B.(ed.), Mineter M.J.(ed.), 1997: Parallel Processing Algorithms For GIS. Taylor&Francis

[3] Huayi Wu, Xuefeng Guan, Jianya Gong, 2011: ParaStream: A parallel streaming Delaunay triangulation algorithm for LiDAR points on multicore architectures. Computers & Geosciences 37(9): 1355-1363

[4] Osterman A., 2012: Implementation of the r.cuda.los module in the open source GRASS GIS by using parallel computation on the NVIDIA CUDA graphic cards. Elektrotechniski Vestnik 79(1-2): 19–24

[5] Soo Hee Han, Joon Heo, Hong Gyoo Sohn and Kiyun Yu, 2009: Parallel Processing Method for Airborne Laser Scanning Data Using a PC Cluster and a Virtual Grid. Sensors 9

[6] Xuefeng Guan, Huayi Wua, 2009: Leveraging the power of multi-core platforms for large-scale geospatial data processing: Exemplified by generating DEM from massive LiDAR point clouds. Computers and Geosciences 36