Parallel Algorithms for Processing Hydrologic Properties from Digital Terrain

R. M. Wallace[1], D.G. Tarboton[2], D.W. Watson[3], K.A.T. Schreuders[4], T.K. Tesfa[5]

[1]Information Technology Laboratory, US Army Engineer Research and Development Center,
3909 Halls Ferry Road, Vicksburg, MS 39180, USA
Email: robert.m.wallace@usace.army.mil

[2] Department of Civil Engineering, Utah State University,
4110 Old Main Hill, Logan, UT 84322-4110, USA
Email: david.tarboton@usu.edu

[3] Department of Computer Science, Utah State University
4205 Old Main Hill, Logan, UT 84322-4204, USA
Email: dan.watson@usu.edu

[4] Department of Civil Engineering, Utah State University
4205 Old Main Hill, Logan, UT 84322-4204, USA
Email: kim.schreuders@usu.edu

[5] Department of Civil Engineering, Utah State University
4205 Old Main Hill, Logan, UT 84322-4204, USA
Email: t.k.t@aggiemail.usu.edu

1. Introduction

A Digital Elevation Model (DEM) represents land surface topography using a rectangular raster grid, where each raster cell contains a floating point value equivalent to the elevation of that geographic point above some base value (usually, sea level) (Wilson and Gallant, 2000). DEMs have become a vital component of the hydrologic modeling process and are used for a number of purposes including distributed hydrologic modelling (Kampf and Burges, 2007) and floodplain mapping (NRC, 2007).

A revolution in the ability to collect elevation data has created a drastic improvement in the quality and quantity of DEM data. Ground resolution of the raster cells has improved from 30-100 meter per raster cell 5-10 years ago to 1-5 meter resolutions today for much of the Earth's land surface. This accuracy has increased the size of the DEMs used for hydrologic purposes. For instance, to represent the Provo River basin in central Utah, $1.73e^5$ hectares (673 mi^2) at 90-meter posting intervals requires $4.7e^5$ raster cells; at 30-meter resolution, $4.2e^6$ cells; and at 10-meter resolution, $3.8e^7$ cells. Because of the increase in raster size, many of the analysis techniques used for coarser resolutions and smaller DEMs are prohibitively time consuming when being applied to high-resolution data.

Although increases in computer processor speed and memory and disk availability have helped enable working with this large data there is a need to adapt hydrologic algorithms to exploit new parallel processing capability and architectural functionality. For example Arge et. al. (2002) examined ways to frame key hydrologic terrain analysis algorithms to take advantage of transparent parallel I/O systems to overcome some of the I/O bottlenecks that occur when processing large terrain datasets in single CPU environments. Arge et al. (2002) implemented single flow direction flow routing (including pit removal) and flow accumulation and showed that efficient algorithms designed to optimize the reading and writing of blocks of data between memory and

disks based on system component properties can significantly improve processing times.

This paper describes parallel algorithms that have been developed to enhance hydrologic terrain processing so that larger datasets can be more efficiently computed. By physically distributing the hydrologic processing for a single dataset among compute nodes in a cluster based system or even a multi core desktop computer, considerable speedup is achieved by simultaneous processing of different portions of the domain. On a cluster based system this approach also takes advantage of the large aggregate memory of all the compute nodes working together. Message Passing Interface (MPI) parallel implementations have been developed for pit removal, flow direction, and generalized flow accumulation methods within the Terrain Analysis Using Digital Elevation Models (TauDEM) package (Tarboton and Baker, 2008; Tarboton et al., 2009).

2. Approach

The parallel algorithms work by decomposing the domain into striped data partitions where each stripe is processed by a separate processor. This method also reduces the memory requirements of each processor so that larger size grids can be processed. An input grid is divided horizontally into equal parts based on the number of processes, with any extra portion remaining being attached to the last partition. Our algorithm stores for each partition the one-cell border information in adjacent partitions and uses a dependency grid to track when these border cells have been evaluated in the adjacent partition and manage iterative swapping of this information using message passing so that meandering across stripes is properly handled. The overhead associated with this iteration, which is input data dependent, is one of the factors that contribute to the parallel algorithms scaling less efficiently than number of processors to the power -1.

Many of the functions in TauDEM are based on the D-infinity (D∞) multiple flow direction model (Tarboton, 1997) that represents flow direction as a vector along the direction of steepest downward slope on eight triangular facets centered at each grid cell. Flow from a grid cell is shared between the two downslope grid cells closest to the vector flow angle based on angle proportioning. Contributing area, defined as the number of grid cells draining through each grid cell, is the simplest example of these functions, and is used to explain the approach. To calculate the contributing area of a cell i, all of the cells in the region that drain into i must first be calculated. A dependency grid that contains at each cell i the number of unevaluated immediate neighbours that drain into i is used to achieve this. If there are no neighbouring cells, n, that drain into i, that cell is considered ready for evaluation, so it is placed on the queue, allowing its contributing area to be calculated. If there is a neighbour that drains into i, the number of neighbours that drain into i is recorded in the dependency grid. This number is used later to determine when cell i will be ready to be calculated. Each process completes this phase in parallel with all other processes.

Each process maintains a queue containing cells that are ready for evaluation and a grid filled with number of cell dependencies. After the queue has been initialized, each process begins popping cells off the queue and calculating each cell's contributing area. Then the dependency grid at each downslope neighbour is decremented by one. If the dependency grid becomes zero, all upslope cells have been calculated and the grid cell is put on the queue. It is possible however that the grid cell may belong to an adjacent process. In this case, instead of decrementing the dependency grid by one, the dependency border is decremented. Upon completion of process queues,

communication between processes is performed to swap border dependency information. The procedure is iterated until all cells have been evaluated.

3. Evaluation

Parallel implementations of the Pit Remove and D∞ Contributing Area functions were tested (Figures 1 and 2) on: (a) a dual quad-core Xeon E5405 2.0GHz PC with 16GB RAM (8 processors); and (b) a cluster of 16 diskless Dell SC1435 compute nodes, each with 2.0GHz dual quad-core AMD Opteron 2350 processors with 8GB RAM (128 processors total) controlled by a similar Dell SC1435 acting as the head node, and housing the file system. Each compute node has a dual-port gigabit network adapter, one of which is reserved for the network file system.

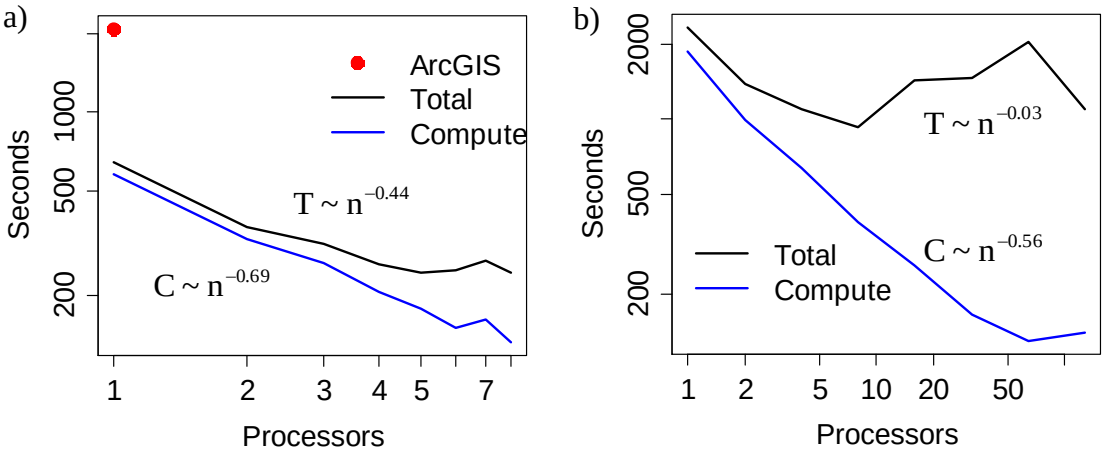

Figure 1. Parallel Pit Remove timing for NEDB test dataset (14849 x 27174 cells ≈ 1.6 GB). a) 8 processor PC, b) 128 processor cluster.

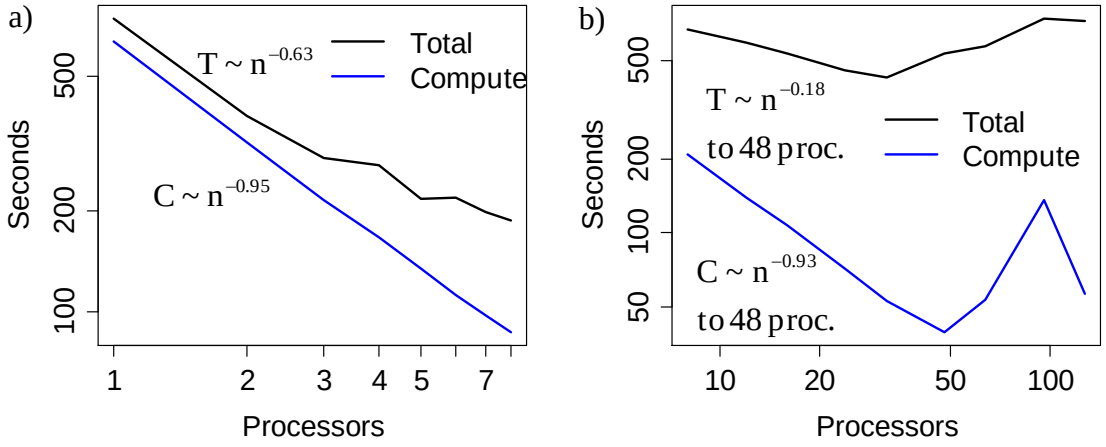

Figure 2. Parallel D-Infinity Contributing Area Timing for Boise River test dataset (24856 x 24000 cells ≈ 2.4 GB). a) 8 processor PC, b) 128 processor cluster.

In the Pit Remove comparison the time taken by the serial ArcGIS implementation of an equivalent Fill function on the 8 processor PC is also shown. On the 128 processor cluster, total time is significantly more than compute time reflecting the cost of reads and writes on this system underscoring the importance of architectures with fast file systems for this class of problems. Both Pit Remove and Contributing Area algorithms appear to be scalable out to about 50 processors with this architecture. This is the point where partition stripe borders occupy about 1/200 of the domain for these data, a

factor likely to be close to the ratio of memory access versus network transfer speeds for data being shared across partitions. These results show that, among other considerations, parallel implementations of hydrologic analysis algorithms must balance increases in execution speed with parallel overhead aspects such as network latency and synchronization costs that are common to all parallel applications.

4. Conclusions

The results demonstrate that the processing times for hydrologic terrain analysis algorithms can be significantly reduced by the use of multiple processor distributed systems. The amount of improvement is dependent upon system architecture and the dataset being evaluated. In an environment where multi-core desktop systems as well as cluster-based systems are becoming more prevalent this approach provides an effective means of increasing analysis capability for a large community of users.

Acknowledgements

This work was funded by the System Wide Water Resources Research Program of the US Army Corps of Engineers.

References

L. Arge, J. Chase, P. Halpin, L. Toma, J. Vitter, D. Urban, and R. Wickremesinghe, 2003, Efficient flow computation on massive grid terrain datasets. Geoinformatica, 7(4):283–313.

S. K. Kampf and S. J. Burges, (2007), "A framework for classifying and comparing distributed hillslope and catchment hydrologic models," Water Resour. Res., 43, http://dx.doi.org/10.1029/2006WR005370

National Research Council Committee on Floodplain Mapping Technologies, (2007), Elevation Data For Floodplain Mapping, The National Academies Press, Washington, DC.

D. G. Tarboton, (1997), "A New Method for the Determination of Flow Directions and Contributing Areas in Grid Digital Elevation Models," Water Resources Research, 33(2): 309-319.

D. G. Tarboton and M. E. Baker, (2008), "Towards an Algebra for Terrain-Based Flow Analysis," in Representing, Modeling and Visualizing the Natural Environment: Innovations in GIS 13, Edited by N. J. Mount, G. L. Harvey, P. Aplin and G. Priestnall, CRC Press, Florida, p.496.

D. G. Tarboton, K. A. T. Schreuders, D. W. Watson and M. E. Baker, (2009), "Generalized terrain-based flow analysis of digital elevation models," 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation, ed. R. S. Anderssen, R. D. Braddock and L. T. H. Newham, Modelling and Simulation Society of Australia and New Zealand and International Association for Mathematics and Computers in Simulation, July 2009, p.2000-2006, http://www.mssanz.org.au/modsim09/F4/tarboton_F4.pdf.

J. P. Wilson and J. C. Gallant, (2000), Terrain Analysis: Principles and Applications. John Wiley and Sons, New York.