A Scalable High-performance Topographic Flow Direction Algorithm for Hydrological Information Analysis

Kornelijus Survila[1,2,3]
survila2@illinois.edu

Ahmet Artu Yıldırım[1,2,3]
ayild@illinois.edu

Ting Li[1,2,3]
tingli3@illinois.edu

Yan Y Liu[1,2,3]
yanliu@illinois.edu

David G Tarboton[4,5]
dtarb@usu.edu

Shaowen Wang[1,2,3]
shaowen@illinois.edu

[1]CyberGIS Center for Advanced Digital and Spatial Studies
[2]National Center for Supercomputing Applications
[3]University of Illinois at Urbana-Champaign, Champaign, IL, USA
[4]Utah Water Research Laboratory
[5]Utah State University, Logan, UT, USA

ABSTRACT

Hydrological information analyses based on Digital Elevation Models (DEM) provide hydrological properties derived from high-resolution topographic data represented as an elevation grid. Flow direction is one of the most computationally intensive functions in the current implementation of TauDEM, a broadly used high-performance hydrological analysis software in hydrology community. Hydrologic flow direction defines a flow field on the DEM that directs flow from each grid cell to one or more of its neighbors. This is a local computation for the majority of grid cells, but becomes a global calculation for the geomorphologically motivated procedure in TauDEM to route flow across flat regions. As the resolution of DEM becomes higher, the computational bottleneck of this function hinders the use of these DEM data in large-scale studies. This paper presents an efficient parallel flow direction algorithm that identifies spatial features (e.g., flats) and reduces the number of sequential and parallel iterations needed to compute their geomorphologically motivated flow direction. Numerical experiments show that our algorithm outperformed the existing parallel D8 algorithm in TauDEM by two orders of magnitude. The new parallel algorithm exhibited desirable scalability on Stampede and ROGER supercomputers.

CCS Concepts

•Computing methodologies → Massively parallel algorithms; •Applied computing → Environmental sciences;

XSEDE '16 July 17–21, 2016, Miami, FL, USA

DOI: http://dx.doi.org/10.1145/2949550.2949571

Keywords

Parallel flow direction assignment; D8 flow algorithm; TauDEM; High-performance computing

1. INTRODUCTION

A grid Digital Elevation Model (DEM) is a common format for digital representation of terrain elevation widely used to extract hydrological and geomorphological information for numerous purposes such as for basin and stream network delineation and wetness index or height above the nearest drainage calculation [20, 16, 14, 9, 7].

Big data challenge the state of the art in computation, network and data storage and will continue in this respect as technology evolves dramatically [10]. Such trend in geographic information science (GIScience) is producing DEMs of much higher resolution, and thus requires highly scalable hydrological information analysis algorithms for handling the size and resolution of big DEM datasets. One example of this data explosion we face is the advent of LiDAR (Light Detection and Ranging) data [8] that are in finer resolution and higher accuracy to represent surface elevation. On the other hand, the capability of handling large DEMs in hydrological analysis provides unprecedented opportunities to looking into grand science problems at larger spatial extent with very fine resolutions.

Hydrological information analysis functions often conduct iterative computation on DEM input, which can be computationally intensive. Flow direction computation is one example. Hydrologic flow direction defines a flow field on the DEM that directs flow from each grid cell to one or more of its neighbors. This is a local computation for the majority of grid cells, but where there are flat areas in the DEM, either where terrain is physically flat, or artificially flat due to hydrologic conditioning by pit filling, this computation may be global. TauDEM uses a geomorphologically motivated procedure due to Garbrecht and Martz [5] to compute flow directions across flats, that directs flow towards adjacent low terrain and away from adjacent high terrain. The present implementation of this is iterative, requires communication across partitions and is computationally intensive. In fact, among more than 30 functions provided in TauDEM [18], an open source parallel computing software for hydrological in-

formation extraction, the two flow direction algorithms (D8 and D-∞) are the most computational expensive functions. In a previous study [4], both of the two functions in TauDEM took more than 40 hours to process a 36GB DEM using 1,024 processors on Stampede supercomputer. Accelerating these two functions has great research significance to enable the TauDEM suite to handle big DEMs for large-scale studies, and thus has been one of the focuses of an awarded XSEDE ECSS project during this academic year.

Drainage direction assignment based on 8 neighbor connectivity (D8 flow algorithm) is utilized extensively in earth science where the flow direction of a DEM grid cell is directed to one of its neighbors with the steepest downwards slope [11, 3]. To alleviate the disadvantage of the D8 flow algorithm that restricts a flow direction into only one of eight possible directions, the D-∞ flow algorithm was introduced [17] in which flow direction is represented as a single angle and flow is proportioned between neighbors based on the angle and angle to adjoining neighbors.

Currently, handling flat regions requires the most computation in TauDEM's flow direction algorithm. Flat regions are not common in nature [5], but their presence in DEMs are due to intentional hydrologic conditioning, involving pit filling [2, 12, 19], so that each grid cell can drain along a continuous path to an outlet or the edge of the domain, as well as a by-product of some computer operations, such as resampling the DEM from higher to coarser resolution [21]. Although finding flow directions on a non flat DEM is trivial, DEMs with flat regions that have no local gradient raise a problem for flow direction assignment and requires iterative computation on its cells [5, 1]. The presence of natural or artificially generated flat regions make it necessary to reconstruct flow direction using efficient algorithms over such areas.

TauDEM incorporates an MPI-based parallel D8 algorithm to facilitate the computation of hydrological flow directions on desktop workstations as well as supercomputing systems [22]. The TauDEM algorithm partitions the data across processors in a stripe fashion, then each processor determines the flow directions using Garbrecht-Martz (G&M) algorithm [5, 16, 20]. Since the flat areas can span across the local partitions of the processors, the algorithm converges to the solution at each iteration where adjacent processors exchange edge or the ghost zone data. However, TauDEM's flow algorithm suffers from low computational efficiency because of excessive communication due to its iterative nature, therefore, finding flow directions over huge DEMs becomes practically nearly impossible even on high-performance computing cluster systems. This inefficiency of TauDEM D8 flow algorithm rapidly exacerbates as the input DEM size increases.

In this study, we present an efficient parallel algorithm for assigning flow directions of cells in big DEMs using the MPI paradigm [6]. In the first phase of the algorithm, processors find the flat regions in a collective fashion using connected-component labeling approach [15]. Once the flat surfaces are determined, each processor resolves its local flats without further communication, or with minimal communication when the flat surfaces reside on a partition boundary of adjacent processors. Finally, each cell on a flat region is assigned a flow direction in an embarrassingly parallel fashion. Experiments show that we significantly outperform the parallel D8 flow algorithm of TauDEM. We believe that the new parallel algorithm will highly impact the dissemination of hydrologic information, especially with the new ability to process big spatial data, such as the entire 10-meter US elevation data, within an hour of computation instead of a few weeks.

2. D8 FLOW ALGORITHM

D8 flow algorithm computes the direction of the flow on a DEM from the source cell to one of its 8 neighbor cells with highest slope. Computing the flow directions using D8 approach is trivial if the elevation of each cell is greater than at least one of its neighbor cells. The workflow of flow direction assignment has four steps, in general. In the first step, the DEM is hydrologically conditioned by filling the depressions/pits [2, 12]. Pits are the cells surrounded by higher elevation cells that no flow can escape. Another option for hydrologic conditioning is carving [13], but TauDEM does not have an implementation for this. Then, a flow direction is assigned to each cell using one of the 8 distinct symbols. If the direction can not be determined, in the that case maximum slope of the cell is zero, then the cell resides on a flat region and is marked with the $FLAT$ symbol. In the third step, connected flat regions are detected where the cells have the $FLAT$ value. Finally, a flow algorithm, e.g. G&M, is performed to determine the final flow directions of the flat cells.

Garbrecht and Martz (G&M) introduced an algorithm to resolve the flow direction assignment problem for DEM cells on flat regions [5]. The algorithm applies the following steps on the cells of flat regions:

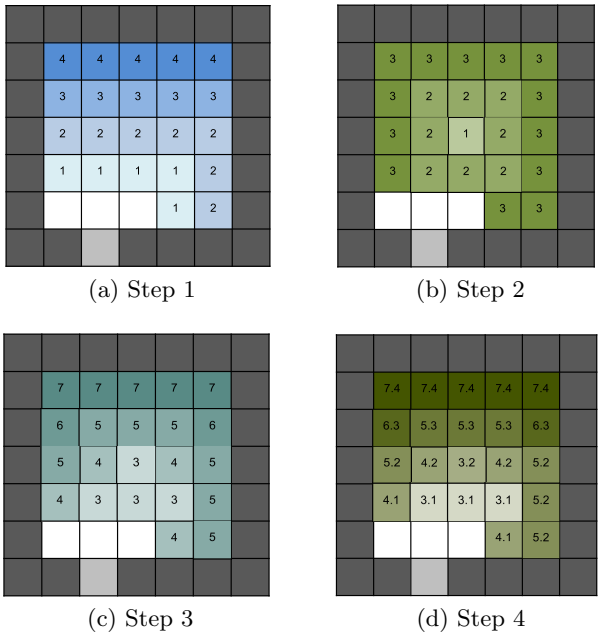
1. Incrementing the elevation by some small value α in an iterative fashion from adjacent lower terrain until each cell has a downstream gradient toward lower terrain (Figure 1a).

2. Incrementing the elevation of the cells by α that are not adjacent to a cell at lower elevation (Figure 1b). This process iterates from outer cells to inside in order to derive the gradient away from the higher terrain.

3. The increments generated in Steps 1 and 2 are added and used to determine the flow direction over the formerly flat region. Equal weight is given to the gradient towards lower terrain and to the gradient away from higher terrain (Figure 1c).

4. If the combination in Step 3 produces new flat regions, a variant Step 1 is repeated on those cells. In this case, the increment used is an infinitesimal value $\beta < \alpha$ where 1/100000 of the vertical DEM resolution is suggested. Step 2 is not repeated. Then, the additional gradient is superimposed on the terrain (Figure 1d).

The current version of TauDEM includes a parallel implementation of the G&M algorithm as the D8 function, however with the variant that instead of step 4 the procedure is recursively iterated to determine the flow direction for flat grid cells. Barnes et al. report that the G&M algorithm resolves flow directions on flat regions in $O(N^{3/2})$ time [1], where N is the number of cells in DEM. Due to the iterative nature of G&M algorithm, the parallelization of G&M needs to exchange neighbor cells during the iterations. Barnes et



(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

Figure 1: G&M Algorithm Steps to determine the flow directions of the cells on a flat region; higher elevation cells are in dark gray; lower elevation cells are in soft gray. ($\alpha = 1$, $\beta = 0.1$)

al. [1] further introduced a new improved algorithm operating in $O(N)$ time by taking advantage of FIFO queues. Barnes algorithm outperforms the G&M algorithm with the improvements made. To our best knowledge, there is no parallel implementation of Barnes algorithm in the literature.

Our new sequential algorithm resolves flats using similar principles as in Barnes' algorithm. The algorithm was proposed and implemented independently prior to our knowledge of Barnes'. There are significant differences in coding, algorithm details and numerical optimization techniques, but they are out of the scope of this paper. This algorithm is a major improvement to the current G&M implementation in TauDEM. In our algorithm, iterations in step 1 and 2 are avoided by expanding the increments and visiting each flat cell only once.

3. PARALLEL D8 FLOW ALGORITHM

The current TauDEM parallel D8 flow algorithm used a direct grid cell level approach to parallelize the G&M algorithm: the code uses an outer loop and an inner loop to iterate the unresolved flats and the elevation increment process (step 1 and 2 in Figure 1), respectively. In the inner loop, MPI collective communication happens during each iteration to retrieve the neighboring cells not located on the same processor. Such parallelization works at cell level and does not need to know flat geometry. It is an easy approach to implement, however it is not efficient and consequently became the major performance bottleneck given that the number of cells and iterations needed can be very large for large DEMs. Also, the synchronization and communication costs increase as more processors are used.

Given the new sequential algorithm, the parallel algorithm was modified to efficiently organize flats and handle flats distributed across processors. We apply a connected-component labeling approach [15] to identify flat regions first. The identified flats are then put into a queue for flat resolving. These flats are categorized into two categories: local and shared. Communication is needed to exchange data among shared flats, but local flats are processed in parallel without further inter-process communication.

Algorithm 1 describes the procedure implemented. First, the input DEM is distributed to each processor evenly in a row-wise fashion. Slope is calculated to determine the flow direction via *CalculateSlope* function. The slope of the flow on the cells/pixels of flat regions is 0, thus they are marked with $FLAT$ symbol. If the flat regions exist on the DEM, the algorithm detects the 8-way connected flat regions named *islands* using a connected component labeling approach in the *findIslands* function. To assign flow directions to the cells on the flat regions, the algorithm first computes drainage of the shared flat regions until there is no change of increment values across all processors. Finally, each processor resolves the local flat regions independently.

Algorithm 1 Parallel D8 flow algorithm

Require: Input elevation DEM D_p of the processor p
Ensure: Flow direction grid F_p of the processor p
1: $F_p \leftarrow$ CalculateSlope(D_p)
2: **if** there exist cells with $FLAT$ value in F_p **then**
3: $IS \leftarrow$ findIslands(F_p)
4: **while** there exists shared flat regions **do**
5: $F_p \leftarrow$ resolveSharedFlats(IS)
6: **end while**
7: $F_p \leftarrow$ resolveLocalFlats(IS)
8: **end if**

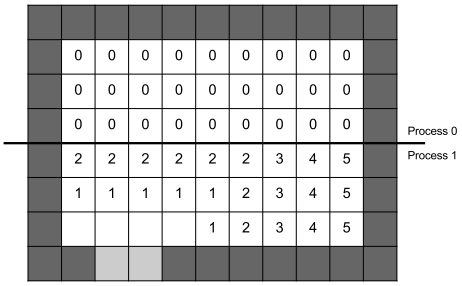
The implemented parallel algorithm requires much less communication among the processors than the current TauDEM implementation as each processor resolves the flow directions for its local flat regions in an embarrassingly parallel fashion. Communication is required to resolve flat regions spanning the partitions of several processors. The steps of the new parallel D8 flow algorithm are described below.

3.1 Step 1: Preprocessing

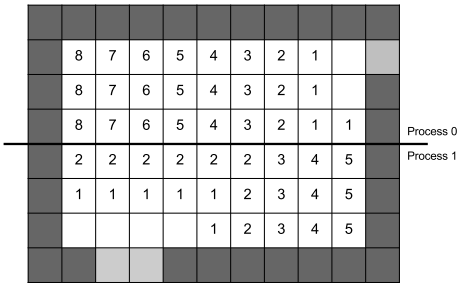
In the parallel approach, all the data are read and written in parallel. A row-wise decomposition is used to divide the whole area into stripes. Each processor handles one partition. Partitions are evenly distributed among the processors, except for the the last process that handles the remainder from the even division of the DEM rows. Note that we have used the existing communication framework that TauDEM already provides. In this framework, at first, each process reads its own rows of data from the DEM file. Since each cell requires neighboring values in the following steps, the calculation of the top or bottom rows in each process also needs neighboring rows (ghost zone data) in other processors. Therefore, neighboring processors exchange the ghost zone with each other.

Each process initially uses the sequential D8 flow direction algorithm, to calculate the flow direction for all the cells in its partition that have positive slope, using information from, but not computing, for the ghost zone.

3.2 Step 2: Identify flat regions



(a) Case 1; a grid with one lower elevation cell on 2 processors



(b) Case 2; a grid with two lower elevation cells on separate partitions on 2 processors

Figure 2: Parallel initialization step of setting gradient towards lower terrain on two cases; higher elevation cells are in dark gray; lower elevation cells are in soft gray

In each process, all the cells without clear flow directions after the initial flow direction assignment are defined as the flat cells. Each cell is also labeled according to the flat area it falls in. Then, all the flat areas are divided into two categories. If a flat area crosses the border of the process, it is defined as a bordering flat and put into a list named *borderingFlatsList*. Otherwise, if it completely falls in the area held by only one process, it is defined as a local flat and should be put into the list of *localFlatsList*. Local flats can be resolved independently, but bordering flats need inter-process communication.

3.3 Step 3: Resolve local flats

The local flats can be processed in each process in exactly the same way as the sequential algorithm, without any communication between processors. A gradient towards lower terrain and a gradient away from higher terrain are both calculated using the new sequential algorithm given above. The final flow direction can be determined through the linear combination of the two gradients. The procedure is repeated until there are flats remaining.

3.4 Step 4: Resolve flat regions bordering other processors

If the flat is shared with another process, communication is inevitable between all the processors that the flat crosses. In this step, all this kind of flats are processed in a slightly different way. Step 4 can be divided into following steps similar to the G&M algorithm: Setting gradient towards lower terrain and setting gradient away from higher terrain.

a. Setting gradient towards lower terrain: The low boundaries of all these flats should be detected first. A flat cell is a low boundary cell if it has at least one adjacent cell at equal or lower elevation with a definite flow direction. All increments of the low boundary cells are set to 1 and then the decremented neighbors of these cells without definite flow direction are incremented to 2. This process continues until no more cells can be incremented. If low boundary can be found, the output value is actually each cell's D8 distance to the outlet. If there is no low boundary cell found, all the cells in the flat area will not have any increment (i.e. equal to 0).

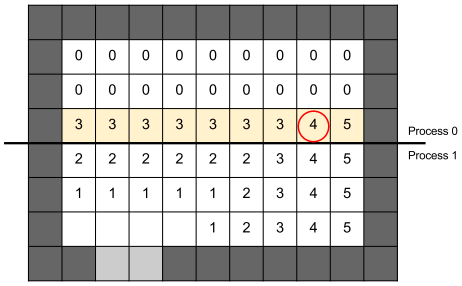


Figure 3: Setting border cells using cells on ghost zone data from process 1; the cell in the red circle has three neighbors with different values (3, 4, 5) and it is assigned a value of 4 which is $3+1$ ($min(N_{ghost})+1$)

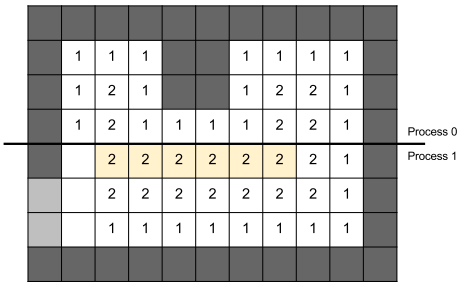
Then, after the local increments are set, each process should communicate with its neighbors to exchange the ghost zone data, if needed. The values of the cells on the border are set to be larger than 0 and less than or equal to the smallest value of its adjacent cells in the ghost zone, i.e., $min(N_{ghost})+1$. This procedure is illustrated in Figure 3.

Once cell values on the border are set, all the neighboring cells of those updated cells should also be checked and updated accordingly using the new border values. If a neighboring cell have no increment or have a higher increment than the value of an updated neighboring cell ($N_{updated}$) plus 1, the increment should be set to $N_{updated}+1$. Otherwise, no change or checking is required. All updated cells are sorted in ascending order based on their increment values, since multiple neighbors of a cell might have lower but different increments. This process is repeated until no more cells need updating.

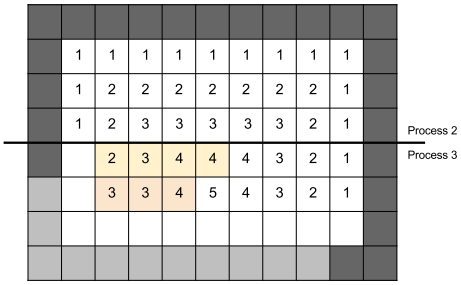
b. Setting gradient away from higher terrain: Similar to the previous step, all high boundary cells of the flat regions should be detected first. A flat cell is a high boundary cell if it has at least one neighbor cell at higher elevation. A local gradient away from higher terrain is also first calculated using the same algorithm as the sequential part. If high boundary can be found, the output value is actually each cell's D8 distance away from higher terrain. If high boundary cannot be found, all the cells in the region will remain unchanged.

After the local increments are set, each process's border values will be updated using the same criteria as step a: all the border values should be equal or lower than $min(N_{ghost})+1$ and also be higher than 0. Then, all the cells neighboring the updated cells should be updated in the same way as step a until no more cells need updating. This step is illustrated in Figure 4.

To compute the final gradient away result, the increments

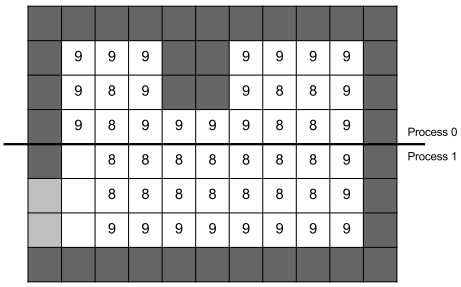


(a) Gradient-away partitions of processors 0 and 1

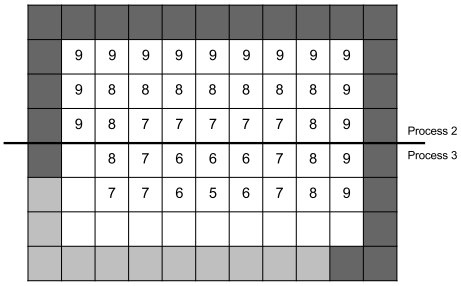


(b) Gradient-away partitions of processors 2 and 3

Figure 4: Setting gradient away from higher terrain in parallel using 4 processors; higher elevation cells are in dark gray; lower elevation cells are in soft gray



(a) Inverted partitions of processors 0 and 1



(b) Inverted partitions of processors 2 and 3

Figure 5: Inverted cell values in setting gradient away from higher terrain using $GlobalMaxIncrement = 10$; higher elevation cells are in dark gray; lower elevation cells are in soft gray

need to be inverted using the global maximum increment of the gradient away from higher terrain, *GlobalMaxIncrement*. *GlobalMaxIncrement* is calculated among all the processors. Each cell's value x should be converted to *GlobalMaxIncrement*-x at the end of Step 4. This operation is illustrated in Figure 5.

Determine the final flow direction: The results from previous 2 steps are also added to get the final elevation increments using an equal weight. The final direction will be assigned over the flats using a D8 flow routing approach in the same way as the sequential algorithm.

4. PERFORMANCE EVALUATION

The new D8 algorithm was evaluated on ROGER supercomputer at the CyberGIS Center and the National Center for Supercomputing Applications (NCSA) and Stampede supercomputer at the Texas Advanced Computing Center (TACC). Each node on ROGER is configured with Intel Xeon E5-2660 processor (2.6GHz, 20 cores/node, 256GB memory). Stampede employs Intel Sandy bridge processor (2.7 GHz, 16 cores/node, 32GB memory).

To compare the performance with existing TauDEM, the most recent TauDEM v5.3.4 with GDAL IO support is deployed with the same compiling options. A DEM of size 429MB with 9,098,177 flats is used to measure the computing time in order to finish the experiment in reasonable amount of time. The experiment was conducted on Stampede using up to 128 processors. Figure 6 depicts the performance comparison in log_2 scale. Using 1 processor, TauDEM v5.3.4 took 4105.27 seconds while the new D8 algorithm took 8.85, demonstrating the effectiveness of the new sequential algorithm. By increasing the number of processors, time taken by TauDEM v5.3.4 decreased to 59.9 seconds using 128 processors, while the new D8 algorithm took 0.23 seconds. On average, the new D8 algorithm was 468 times faster than the D8 in TauDEM v5.3.4 for this DEM. The dramatic performance improvement mainly attributes to the improved numerical efficiency of the sequential flat resolving algorithm and the resulted reduction of communication by the parallel algorithm. Increasing the number of processors from 1 to 128, the percentage of the shared flats increased from 0% to 42.16%. But since the required communication was minimum, such increase did not result in significant increase of communication cost. Therefore, the execution time continued to drop to 0.23 seconds. It is worth noting that using 256 processors, the new D8 algorithm took longer than using 128 processors, just because the workload became small enough compared to the overhead in IO and communication, which indicates that 256 processors is an overkill for processing this DEM.

Figure 7 shows the performance details of the new D8 algorithm on ROGER for processing a larger DEM - the Chesapeake DEM of size 11.3 GB with 2.8 billion cells and 578.9 million flats. 1 to 160 processors were used to understand the cost of each component in the algorithm. Since flat resolving dominates the execution time, Figure 7 only shows the time distribution of the three components in flat resolving: flat identification, resolving shared flats, and resolving local flats. The sequential flat resolving took 425.26 seconds to finish, while using 160 cores (8 nodes) took 29.42 seconds. When the number of processors is between 20-50, variation in flat identification was observed. The reason is still being investigated. When using more than 100 proces-

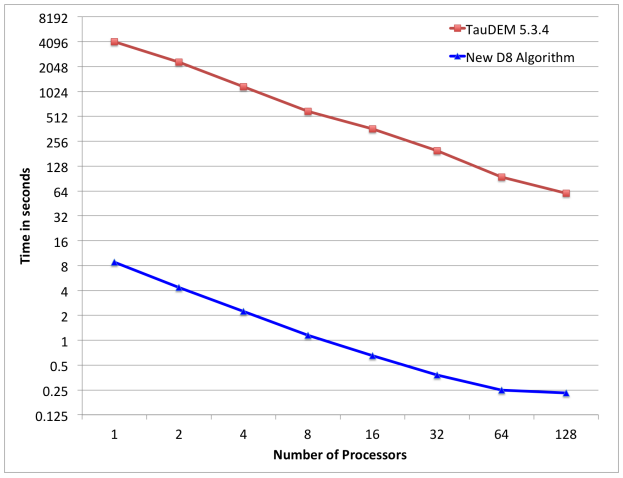


Figure 6: Performance comparison with D8 in TauDEM 5.3.4 on Stampede. Plot is in log_2 scale. DEM size: 429MB (10812x10812 cells). Number of flats: 9,098,177.

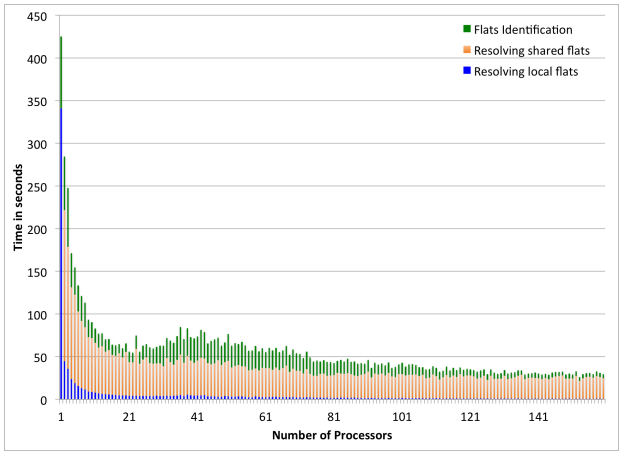


Figure 7: Performance details of the new D8 algorithm on the 11.3GB (53248x53248 cells) Chesapeake DEM on ROGER. Number of flats: 578,895,436.

sors, we did not observe significant performance improvement, mainly because there was little variation on the number of shared and local flats per process. Resolving shared flats, since communication is involved, took majority of the flat resolving time. Resolving local flats was fast. On the other hand, the best D8 performance among all previous TauDEM versions for the Chesapeake DEM on ROGER is 4933.69 seconds in total time and 4698.69 seconds for flat resolving, obtained using TauDEM v5.2.1 multifile version and 500 processors (25 nodes).

To test the performance of the new D8 algorithm, we chose two large hydrologically conditioned DEMs at the U.S. watershed region (HUC2) level to test on Stampede using 256 processors. The compressed DEMs of the two HUC2 regions are of size 33GB (HUC 11) and 67GB (HUC 10), respectively. Table 1 shows the DEM size, number of flats, and computing time. Previous TauDEM versions were not able to calculate flow direction for such size DEMs. For the Arkansas-White-Red region, 55.63% flats were identified as local flats and processed without inter-process communication in 3.70 seconds, while the 44.37% shared flats took 32.03 seconds for processing. For the Missouri region, 66.22% flats were identified as local flats and processed locally in 12.09 seconds, while the 33.78% shared flats took 68.79 seconds for processing.

Table 1: Performance on large watershed DEMs (Time in seconds).

Region (HUC2)	Size	Num Flats	Time
Arkansas-White-Red (11)	170919x114734	433,807,730	39.57
Missouri (10)	227191x177787	1,225,082,919	91.43

5. CONCLUSIONS

TauDEM introduced the first parallel D8 flow algorithm towards solving the direction assignment problem based on the sequential G&M algorithm. In this paper, we significantly improved the numerical performance of the existing D8 algorithm. First, we designed an efficient sequential algorithm that removes the expensive inner loop in the original algorithm. Second, the parallel algorithm identifies flat regions and uses the locality information of those flats to coordinate the communication and communication for local and shared flats. The flat regions which are local to the processor are resolved without communication. Experiments show that then enhanced parallel D8 algorithm significantly outperforms the current TauDEM version. Before, it took days for D8 to process a 36GB DEM. Now we are able to process a 67GB compressed DEM within two minutes. The performance acceleration achieved in our work will greatly enhance TauDEM's capability of handing big DEM datasets at large spatial extent, e.g., at national level, and enable the computational hydrology community to conduct hydrological analyses that depend on D8 computation to unprecedented scales.

We will further test the performance of the new algorithm using larger DEMs and study the impact of spatial data characteristics on algorithm performance, e.g., the impact of flat size, distribution, and density in relation to spatial extent. The same flat resolving and parallelization strategy will be applied to the D-∞ flow direction function in TauDEM. The new algorithm will then be integrated into TauDEM for a new release on GitHub.

6. ACKNOWLEDGMENTS

This work is part of the ECSS project (award number EAR130008) of XSEDE, which is supported by National Science Foundation (NSF) grant number ACI-1053575. This material is based in part upon work supported by the NSF under grant number 1047916. The work used the ROGER supercomputer, which is supported by NSF under grant number: 1429699. We also acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources on the Stampede system that have contributed to the research results reported within this paper.

7. REFERENCES

[1] R. Barnes, C. Lehman, and D. Mulla. An efficient assignment of drainage direction over flat surfaces in

raster digital elevation models. *Computers & Geosciences*, 62:128 – 135, 2014.

[2] R. Barnes, C. Lehman, and D. Mulla. Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *Computers & Geosciences*, 62:117–127, 2014.

[3] J. Fairfield and P. Leymarie. Drainage networks from grid digital elevation models. *Water Resources Research*, 27(5):709–717, 1991.

[4] Y. Fan, Y. Liu, S. Wang, D. Tarboton, A. Yildirim, and N. Wilkins-Diehr. Accelerating taudem as a scalable hydrological terrain analysis service on xsede. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, page 5. ACM, 2014.

[5] J. Garbrecht and L. W. Martz. The assignment of drainage direction over flat surfaces in raster digital elevation models. *Journal of Hydrology*, 193(1–4):204–213, 1997.

[6] W. Gropp, E. Lusk, and R. Thakur. *Using MPI-2: Advanced features of the message-passing interface*. MIT press, 1999.

[7] A. T. Haile and T. Rientjes. Effects of lidar dem resolution in flood modelling: a model sensitivity study for the city of tegucigalpa, honduras. *ISPRS WG III/3, III/4*, 3:12–14, 2005.

[8] X. Liu, J. Peterson, and Z. Zhang. High-resolution dem generated from lidar data for water resource management. In *Proceedings of the International Congress on Modelling and Simulation (MODSIM05)*, pages 1402–1408. Modelling and Simulation Society of Australia and New Zealand Inc., 2005.

[9] X. Liu and Z. Zhang. Drainage network extraction using lidar-derived dem in volcanic plains. *Area*, 43(1):42–52, 2011.

[10] C. Lynch. Big data: How do your data grow? *Nature*, 455(7209):28–29, 2008.

[11] J. F. O'Callaghan and D. M. Mark. The extraction of drainage networks from digital elevation data. *Computer vision, graphics, and image processing*, 28(3):323–344, 1984.

[12] O. Planchon and F. Darboux. A fast, simple and versatile algorithm to fill the depressions of digital elevation models. *Catena*, 46(2):159–176, 2002.

[13] P. Soille, J. Vogt, and R. Colombo. Carving and adaptive drainage enforcement of grid digital elevation models. *Water Resources Research*, 39(12):n/a–n/a, 2003. 1366.

[14] P. J. Soille and M. M. Ansoult. Automated basin delineation from digital elevation models using mathematical morphology. *Signal Processing*, 20(2):171 – 182, 1990.

[15] G. Stockman and L. G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[16] D. Tarboton, K. Schreuders, D. Watson, and M. Baker. Generalized terrain-based flow analysis of digital elevation models. In *Proceedings of the 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation, Modelling and Simulation Society of Australia and New Zealand and International Association for Mathematics and Computers in Simulation*, pages 2000–2006, 2009.

[17] D. G. Tarboton. A new method for the determination of flow directions and upslope areas in grid digital elevation models. *Water resources research*, 33(2):309–319, 1997.

[18] D. G. Tarboton. *Terrain Analysis Using Digital Elevation Models (TauDEM)*. Utah Water Research Laboratory, Utah State University, 2016. http://hydrology.usu.edu/taudem/.

[19] D. G. Tarboton, R. L. Bras, and I. Rodriguez-Iturbe. On the extraction of channel networks from digital elevation data. *Hydrological Processes*, 5(1):81–100, 1991.

[20] T. K. Tesfa, D. G. Tarboton, D. W. Watson, K. A. Schreuders, M. E. Baker, and R. M. Wallace. Extraction of hydrological proximity measures from {DEMs} using parallel processing. *Environmental Modelling and Software*, 26(12):1696 – 1709, 2011.

[21] J. Vaze, J. Teng, and G. Spencer. Impact of dem accuracy and resolution on topographic indices. *Environmental Modelling and Software*, 25(10):1086–1098, 2010.

[22] C. Wallis, D. Watson, D. Tarboton, and R. Wallace. Parallel flow-direction and contributing area calculation for hydrology analysis in digital elevation models [c]. In *PDPTA'09, The 2009 International Conference on Parallel and Distributed Processing Techniques and Applications*, 2009.