Extraction of drainage network from grid terrain datasets using parallel computing

Ling JIANG
Nanjing Normal University
Nanjing, China
jinagling_xs@163.com

Guo-an TANG
Nanjing Normal University
Nanjing, China
tangguoan@njnu.edu.cn

Kai LIU
Nanjing Normal University
Nanjing, China
liukaigiser@163.com

Jian-yi YANG
Nanjing Normal University
Nanjing, China
yang_gis@163.com

Abstract—**Drainage-network extraction from grid terrain datasets is the essential content for hydrologic analysis, soil erosion and geomorphology. However, drainage-network extraction is normally very time-consuming using the sequential program, especially for grid terrain datasets with high resolution and large scope. This paper proposed a set of parallel algorithms to preprocess DEM, determine flow routing and delineate drainage network using cluster computers in an MPI programming model. Both depressions and flat areas could be processed by the DEM preprocessing algorithm. For using the number of upslope-dependence neighbors of a cell, the flow-routing algorithm makes flow accumulation calculation similar to a local algorithm. The experiment results shows that the proposed parallel approach to extract drainage-network performs much better than sequential algorithms and has the better parallel efficiency.**

I. INTRODUCTION

As one of the most significant spatial data, digital elevation models (DEMs) are data structures representing rectangular grids of terrain data composed of cells arranged as a raster, where the value of each cell denotes the elevation of that geographic point above some base value [1]. Based on DEMs, it could mine much useful information about the morphology of a terrain surface via the technology of digital terrain analysis (DTA). The drainage network derived from DEMs is one of the key contents in DTA and has a marked influence on many fields, such as hydrologic analysis, soil erosion and geomorphology [2, 3]. Commonly, drainage-network delineation from DEMs is based on a model for representation of flow routing across the surface and between grid cells [4]. To achieve flow routing, it need to firstly determine a flow direction to each grid cell, and then calculate flow accumulations. According to the flow allocation ratio from a cell when it drains into the neighboring cells, existing flow-direction algorithms can be classified into two main types: single-flow-direction (SFD) algorithms [4, 5] and multiple-flow-direction (MFD) algorithms [6, 7]. Among all the flow-direction algorithms, the D8 algorithm proposed by O'Callaghan and Mark (1984) is widely applied for its simplicity and efficiency [8,9]. From the flow direction grid, a flow accumulation grid can be achieved, and then drainage network can be extracted by means of a threshold [4].

During drainage-network extraction for real DEMs, a DEM preprocessing algorithm is normally applied to fill in the depressions and remove the flat areas in the DEMs before calculating the flow direction [2]. These depressions and flat areas which are both from real topographic features and errors introduced during the DEM production process would cause to fail to obtain correct result for flow direction. Over the last twenty years, many DEM preprocessing algorithms have been develop to obtain a smooth DEM without any depression and flat area [10, 11]. The algorithm proposed in [11] has generally been recognized to perform better from the perspective of algorithm efficiency [12].

At present, along with the development of spatial data capture technology, DEMs have increasingly been big datasets for large scopes and high resolutions [13]. Meanwhile, drainage-network extraction is normally very time-consuming caused not only from the recursive routing algorithm, but also from the iterative DEM preprocessing algorithm. As a result, the traditional serial algorithm for calculating drainage basin is usually hard to meet users' demand of time responding for its long execution time. To speed up the execution time, some researchers have proposed parallel solution for drainage-network extraction [14-16]. These researches focused on the segment of drainage-network extraction, however, DEM preprocessing phase, by far the correction procedure and time consuming, was not processed in parallel. Wallis [17] and Song [18] proposed a parallel DEM preprocessing algorithm from the method of Planchon and Darboux, respectively. However, these parallel algorithms could only fill the sinks to flat areas, and could not remove the flat areas with a very small slope gradient.

This paper presents a design and implementation of parallelized drainage-network extraction. In this work, it adopts Planchon and Darboux's method for DEM preprocessing and the

D8 approach for flow routing to parallelize algorithms using MPI. The remainder of this paper is organized as follows. In section 2, we introduce the implementations of both serial algorithms and parallel ones. In section 3, the performance results from parallel computing are compared with sequential computing on a small cluster, and in Section 4, the main conclusions are drawn.

II. COMPUTATION ALGORITHMS FOR EXTRACTING DRAINAGE NETWORK

The drainage network is a basic feature of terrain surface. Extracting drainage network from DEMs commonly contains three steps: (1) DEM preprocessing, which fills dispersions and removes flat areas, (2) flow routing, which determines the flow direction and flow accumulation for each grid cell, and (3) drainage-network delineating, which identifies a grid cell as being part of a channel using a fixed threshold. Among them, the steps of DEM preprocessing and flow routing are main parts for drainage-network extraction. They are usually the iteration processes and very time consuming. When the first two steps implemented, a common approach for the last step is to decide a cell if its flow accumulation is greater than a fixed threshold [4].

A. DEM preprocessing

The DEM preprocessing algorithm proposed by Planchon and Darboux (called P&D algorithm for short) includes two stages: (1) the water-covering stage, which inundates a thick layer of water over the entire DEM except for the boundary, namely cells on boundary initialized values of the original DEM and the remain initialized an infinite height, and (2) the water-removing stage, which drains the excess water to ensure that for each cell, there is a path that leads to the boundary [11]. Algorithm 1 is the pseudocode of the P&D algorithm. Line 1 implements the water-covering stage and the intensive computing part of the algorithm is the water-removal stage, an iterative process (lines 2-18).

Algorithm 1 Pseudo code of sequential P&D DEM preprocessing algorithm. (zDEM and wDEM are the input DEM and the output result respectively, gap is the very small slope gradient.)

```
viod sDEMProcessing(zDEM, wDEM, gap)
1:  wDEM = Watercovering(zDEM)
2:  do
3:    finished = true
4:    for each c in wDEM do
5:      if wDEM(c) > zDEM(c) then
6:        min_neighbor ← minimum value of neighbors of c
7:        if zDEM(c) >= min_neighbor + gap then
8:          wDEM(c) = zDEM(c)
9:          finished = false
10:       else
11:         if wDEM(c) > min_neighbor + gap then
12:           wDEM(c) = min_neighbor + gap
13:           finished = false
14:         end if
15:       end if
16:     end if
17:   end for
18: loop while not finished
```

B. Flow routing

The D8 (deterministic eight-node) algorithm, proposed by O'Callaghan and Mark, directs flow from each grid cell to one of eight neighbors based on slope gradient [4]. The simplest method of calculating flow direction is to determine the slope (Si) to each neighbor and set it to the direction for which Si is the greatest [3]. For each cell, there are two steps to complete the calculation: computing the maximum gradient of eight neighbors and encoding the flow direction to the cell as Figure 1.

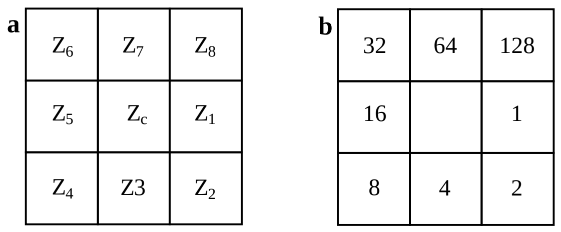

Figure 1. Three-by-three subgrid for the D8 algorithm calculation: (a) Node numbering convention. (b) Flow direction encoding convention.

Once the flow direction is achieved, it could be applied to calculate flow accumulation for each cell. For a cell, the flow accumulation means the flow of all upstream cells which flow into the cell. Assuming each cell has one unit of water, its flow accumulation is the water of all neighbors which drain into it plus their own water (i.e., one multiplies their number):

$$A(c) = n + \sum_{D} A(D) \tag{1}$$

where c denotes any grid cell, n is the number of neighbors flowing into c and D is each neighbor of c whose flow direction is towards c. To calculate the flow accumulation of a cell c, all cells in the region that drain into c must first be calculated [17]. It is means that a cell could achieve its flow accumulation only after all the neighbors which drain into the cell implement their calculation. Assuming m is the number of neighbors that drain into cell c, the flow accumulation of c could be calculated when m equals to 0. Along with m of each cell gradually increased to 0, the flow-accumulation calculation would be done.

The sequential flow-routing algorithm is shown in Algorithm 2. Lines 1-8 serve to achieve the flow-direction grid. In lines 9-20, an iteration process is used for flow accumulation calculation. It is no doubt that the time consuming of flow accumulation computation is much more than that of flow direction.

Algorithm 2 Pseudo code of serial flow-routing algorithm. (wDEM is the input DEM after preprocessing , FDIR and FACC are the output flow direction and flow accumulation.)

```
viod sFlowRouting (wDEM, FDIR, FACC)
1:          for each c in wDEM do
2:             if c is not boundary of wDEM then
3:                i ← the greatest slope between c and neighbor i
4:                FDIR (c) = 2 ^ (i-1)
5:             else
6:                FDIR (c) = NODATA
7:             end if
8:          end for
9:          do
10:      finished = true
11:      for each c in FDIR do
12:         if flow accumulation of c is not calculated then
13:            m ←number of neighbors that drain into cell c
14:            if m == 0 then
15:               FACC(c) ← accumulation calculation by Eq. 1
16:               finished = false
17:            end if
18:         end if
19:      end for
20:   loop while not finished
```

C. *Parallel computation*

The first requirement to extract drainage network in parallel is to partition the data among processors. The paper uses a method proposed by Wallis [17], namely, striped partitioning scheme. In this approach, the input dataset is divided horizontally into *size* equal partitions to *size* processors, with any extra portion remaining being attached to the last subdomain. Each process reads in its assigned partition from a grid file, along with an attached buffer which consists of a row of cells as overlapping area from adjacent partitions above and below it. The buffers could be easily updated by obtaining information from adjoining partitions. From these buffers, each processor has access to neighboring partitions with little communication.

According to the above analysis, the parallel algorithms that integrate single program, multiple data (SPMD) are developed. Algorithm 3 and Algorithm 4 are the parallel programs for drainage-network extraction step1 and step 2 respectively, along with Algorithm 5 for step 3. In these parallel algorithms, the function *ShareBuffers( )* severs to update the buffers of each partition - the last row of processor P_i is copied to the top buffer of processor P_{i+1} and the top row of processor P_{i+1} is copied to the bottom buffer of processor P_i. In the function *RingTermination( )*, the final signal gets 0 if any processor is 0, in addition, the final signal gets 1 while all processors are 1, here, 0 denotes an incompletion status. For Algorithm 3, the function *Waterremoving( )* could find its implementation in lines 3-17 of Algorithm 1. And in line 4 of Algorithm 4, the function *Flowaccumulation( )* is completed as lines 10-19 in Algorithm 2.

Algorithm 3 Pseudo code of parallel P&D DEM preprocessing algorithm. (zDEM and wDEM are the input DEM and the output result respectively, gap is the very small slope gradient.)

```
viod pDEMProcessing(zDEM, wDEM, gap)
1:    wDEM = Watercovering(zDEM)
2:    do
3:       ShareBuffers(wDEM)
4:       finished = Waterremoving(zDEM, wDEM, gap)
5:       finished = RingTermination(finished)
6:    loop while not finished
```

Algorithm 4 Pseudo code of parallel flow-routing algorithm. (wDEM is the input DEM after preprocessing , FDIR and FACC are the output flow direction and flow accumulation.)

```
viod pFlowRouting(wDEM, FDIR, FACC)
1:    FDIR = Flowdirection(wDEM)
2:    do
3:       ShareBuffers(FACC)
4:       finished = Flowaccumulation(FDIR, FACC)
5:       finished = RingTermination(finished)
6:    loop while not finished
```

Algorithm 5 Pseudo code of parallel network-delineating algorithm. (FACC and DNET are the output flow accumulation and drainage network, respectively.)

```
viod pNetwork Delineating (FACC, DNET, threshold)
1:    for each c in FACC do
2:       if FACC(c) > threshold then
3:          DNET (c) = 1
4:       else
5:          DNET (c) = 0
6:       end if
7:    end for
```

III. EXPERIMENTS AND RESULTS

The parallel algorithms proposed in this paper has been implemented in C++ using MPI library and tested on a 10-node cluster linked with a Gigabit Ethernet network. These nodes have the same hardware configuration with Intel Core Quad-Core i3-2350 2.3GHz with 4 GB Memory and operating system. For comparison, the sequential algorithms were executed on a single processor within the cluster. Two datasets on a hilly area of Loess Plateau are chosen as study area, the first one consists of a large grid with size 4589 by 6832, and the second dataset consists of a small grid with size 688 by 1025. The time taken to complete the extraction task was measured using the serial algorithms for each dataset. For the parallel algorithms, the time taken to complete the task was measured using a varying number of processors. Figure 2 shows the total execution time of different processors for drainage-network extraction for the small dataset, while Figure 3 provides the total execution time for the large dataset.

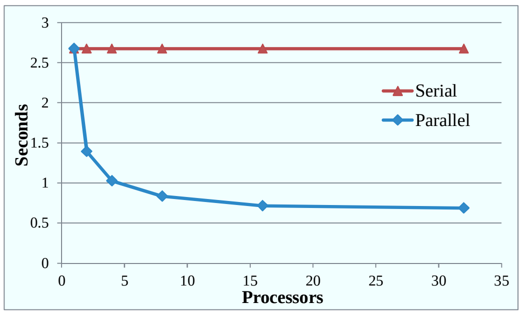

Figure 2. Time taken to complete drainage-network extraction on the small DEM

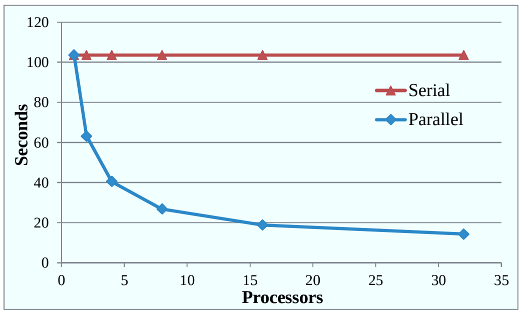

Figure 3. Time taken to complete drainage-network extraction on the large DEM

As seen in Figure 2 and Figure 3, the advantage of parallelization for drainage-network extraction is apparent with the crossover at two processors. The computing time within the two processors is obviously shorter than that of the sequential algorithm. Compared with the serial mode, the parallel results of the large and small datasets are almost identical in terms of increased performance speed, even though the data volumes of the large and small datasets are significantly different from each other. With the increase of processor number, the computational time of the parallel implementation decreases greatly with the crossover at eight processors as the increasing of communication time. Fortunately, the proportion of communication is small and the phase scaled well. The parallel algorithm of drainage-network extraction with a large dataset can achieve good performance acceleration. However, the parallel performance does not increase further when the dataset volume achieves a certain size.

IV. CONCLUSIONS

In this paper, we present a set of parallel algorithms for drainage-network extraction from grid terrain datasets. Compared with the existed algorithms, the parallel DEM preprocessing algorithm proposed in the paper could not only fill the depressions, but also remove the flat areas. And the parallel flow routing algorithm makes flow accumulation calculation similar to a local algorithm as using the number of upslope-dependence neighbors of a cell. Furthermore, the parallel strategy is relatively simple and can be easily augmented for other similar terrain analysis parameters.

Although this research deals specifically with drainage-network extraction, the drainage-basin delineation is a directly extensional concept for further study, and could be easily implemented based on this paper.

ACKNOWLEDGMENT

This work was supported by the National High Technology Research and Development Program of China (2011AA120303), the National Natural Science Foundation of China (40930531, 41171298) and the Foundation of Graduate Innovation Plan of Jiangsu Province under Grant (CXZZ12_0391).

REFERENCES

[1] Wilson, J.P., and Gallant, J.C., 2000. Terrain Analysis: Principles and Applications. John Wiley and Sons, New York.

[2] Hengl, T., and Reuter, H.I., 2008. Developments in soil science, In: Geomorphometry: Concepts, Software, Application, vol.33. Elsevier, Amsterdam, Netherlands, 707pp.

[3] Wilson, J.P., and Gallant, J.C., 2000. Terrain Analysis: Principles and Applications. Wiley, New York, pp. 52-55.

[4] O'Callaghan, J.F., and Mark, D.M., 1984. The extraction of drainage networks from digital elevation data. Computer Vision, Graphics, and Image Processing, 28(3):323-344.

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

[6] Freeman, T.G., 1991. Calculating catchment area with divergent flow based on a regular grid. Computers & Geosciences, 17(3):413-422.

[7] Qin, C.Z., Zhu, A.-X., Pei, T., Li, B.L., Zhou, C., and Yang, L., 2007. An adaptive approach to selecting a flow-partition exponent for a multiple-flow-direction algorithm. International Journal of Geographical Information Science, 21(4):443-458.

[8] Moore, I.D., Hydrologic Modeling and GIS, In Gooldchild, M.F., Steyaert, L.T., Parks, B.O., Crane, M.P., Maidment, D.R., and Glendinning, S. (Eds.), 1996. GIS and Environmental modeling: Progress and Research Issuse, GIS World Books, Fort Collins, pp.143-148.

[9] Tarboton, D.G., Schreuders, K.A.T., Watson, D.W., and Baker, M.E., 2009. 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, Cairns, Australia, pp. 2000-2006.

[10] Jenson, S.K., and Domingue, J.O., 1988. Extracting topographic structure from digital elevation data for geographical information system analysis. Photogrammetric Engineering and Remote Sensing, 4(11):1593-1600,

[11] Planchon, O., and Darboux,F.A., 2001. fast, simple, and versatile algorithm to fill the depressions of digital elevation models. Catena, 46(2-3):159-176.

[12] Wang, L., and Liu, H., 2007. An efficient method for identifying and filling surface depressions in digital elevation models for hydrologic

analysis and modelling. International Journal of Geographical Information Science, 20(2):193-213.

[13] Hengl, T., and Reuter, H.I., 2009. Geomorphometry concepts, software, applications. In: A.E. Hartemink and A.B. McBratney (Eds.), Developments in Soil Science, first ed. Elsevier, Amsterdam, p.765.

[14] Gong, J., and Xie, J., 2009. Extraction of drainage networks from large terrain datasets using high throughout computing. Computers & Geosciences, 35(2):337-346.

[15] Ortega, L., and Rueda, A., 2010. Parallel drainage network computation on CUDA. Computers & Geosciences, 36(2):171-178.

[16] Mower, J.E., 1993. Implementing GIS procedures on parallel computers: a case study. In: Proceedings of the Eleventh International Symposium on Computer-Assisted Cartography, Minneapolis, MN,USA, pp.424-43.

[17] Wallis, C., Wallace, R., Tarboton, D.G., Watson, D.W., Schreuders, K.A.T., and Tesfa, T.K., 2009. Hydrologic Terrain Processing Using Parallel Computing, 18th World IMACS/MODSIM Congress, Cairns, Australia, 13-17 July.

[18] Song, X.D., Tang, G.A., Jiang, L., Zhang, G., and Qian, K.J., 2012. A Novel Parallel Depression Removing Algorithm for Hydrology Analysis in Digital Elevation Models, 20th International Conference on Geoinformatics, Hong Kong, China, 15-17 June.