Parallel Algorithm for Calculating Cost Distance of Raster Data

Y. Wang, C. J. Li, X. B. Yan, J. Wang

School of Computer Science, China University of Geosciences, 430074
Telephone: 13808669081, 15827030409, 13618603396, 15527495676
Fax: -
Email: giswy@126.com, cuglicj@126.com, 814443137@qq.com, jjseen@163.com

1. Introduction

Nowadays, the ever-growing amount of information in spatial databases makes Geographic Information Systems (GIS) based on the traditional sequential models may take excessive time to complete jobs. Fortunately, distributed computing environment is readily available in recent years (Lin Snyder 2009). Consequently, parallelism is becoming a reasonable solution for this dilemma. MapReduce (Wang K Han JZ Tu BB Dai J Zhou W 2010) ,a distributed parallel processing model and execution environment that proposed by Google, can process large data sets running on large clusters of commodity machines. Hadoop (Dean Ghemawat 2008), an open-source software of MapReduce with Hadoop Distributed File System (HDFS), is easy for parallel GIS algorithms because its run-time system takes care of the messy details of parallelization, fault-tolerance, locality optimization, inter-machine communications and load balancing (Wang K Han JZ Tu BB Dai J Zhou W 2010). Though there are still many problems in detail, realizing parallel GIS at a low cost based on these technologies is feasible. Now, obtaining parallel GIS algorithms is a meaningful work.

Distance is critical variable in many geographic analyses (Eastman 1989). A common requirement of raster-based Geographic Information Systems is the determination of distance. For instance, if the distances from each grid cell to the nearest designated feature one can be calculated, a buffer zone of any given distance may then be established for raster data. Moreover, knowledge of distance is also essential when resources are clustered, and the type or level of activity that may be maintained is consequently distance-dependent. For example, the difference between animal species in the importance of distance to the nearest well is an important consideration in range management (Olsson L 1985). Likewise, distance is a common ingredient in the assessment of processes that exhibit distance decay, including processes of mineralization, locational economics, and assessments of risk (Eastman 1989).

There are some algorithms for calculating cost distance of raster data which defines the cost of each cell to its nearest feature cell. In fact, some of them are used in various softwares. However, to meet the demand of high-speed proceeding large scale image, parallel processing is very necessary for the classical algorithms. It is the main motivation of this work.

In this work, a parallel algorithm for calculating cost distance based on raster data is proposed. This algorithm can reduce the time for computing distance if it runs on a parallel computing platform which is meeting requirements. After that, algorithm analysis is made.

2. Related researches

Traditional algorithms are introduced in literature. They are exhaustive search (Olsson L 1985), growth rings (Tomlin CD 1986) and pushbroom (Tomlin CD 1983 1986), etc. Our algorithm is inspired by them.

Exhaustive search is the earliest one. To calculate the nearest distance form any cell to feature ones, this approach uses row and column subscripts to calculate the Euclidian distance from current grid cell to each feature one using the Pythagorean Theorem (Olsson L 1985). This approach is easy to understand and implement. However, with the increasing of raster image size, the computational complexity increases exponentially. To process a raster image with n cells, the computational complexity is O(n2). This makes this approach an unrealistic solution for large raster data sets (Kang 2011). Nevertheless, for a parallel algorithm, only a little sub-image is processed in every computing process. Using the Pythagorean Theorem to obtain the distance is the most convenient way in such a scenario.

In both growth rings and pushbroom, distance of a new cell is computed by that of its neighbors. Differently, pushbroom records square distance of each cell instead of distance itself. This idea can avoid frequent extraction of square root operation and accumulation of rounding errors aroused by that.

3. Parallel Algorithm

To realize the parallel algorithm for calculating cost distance of raster data, some steps are taken in this paper. Firstly, image is divided into sub-images; Secondly, cells in public edges of sub-images are assigned attribute value; Thirdly, distance of each cell in each sub-image is get according to the feature cells in this sub-image; Fourthly, distance is corrected by taking all outside feature cells into consideration; Finally, All sub-images are merged. These steps are described in detail as below.

3.1 Image dividing

Let the source image be m×n cells. It should be divided into some k×k cells sub-images. If n%k==0&&m%k==0, the source image can be exactly divided. Otherwise, the image should be expanded to make m and n meeting the condition. Sub-images have ordinal numeration and are send to corresponding computing processes.

3.2 cells in edges of sub-images are assigned attribute value

One cell in the public edges has its attribute value besides its distance. Let the global coordinate of the nearest feature cell be (X_p, Y_q) and the global coordinate of current cell be (X_n,Y_m). This attribute value is (X_n-X_p, Y_m-Y_q). To find the nearest feature cell, the distance to every feature cells must be compared.

3.3 Computing distance according to the feature cells in local sub-image

In this step, distance is computed by the Pythagorean Theorem. At the same time, square distance of each cell is recorded. It can be seen that the advantage of exhaustive search and pushbroom are both taken into consideration.

3.4 Correcting distance according to the outside feature cells

Each cell in an edge has a attribute value. The coordinate of the nearest feature cell for them can be computed according to the attribute value. If the feature cell is out of this sub-image, its coordinate will be recorded. After the computation for every cell in an edge, the coordinate of all the outside feature cell that possibly influence the distance of the cells in this sub-image are found. Then, every distance in this sub-image is corrected by each found outside feature cell.

3.5 Getting whole image

All sub-images are merged into a whole image one by one according to their number. Then, the finial result is obtain.

3.6 Pseudocode

```
if ID is master then
  count_of_feature_pixel := 0
  count := 0
  for i := 0 to row - 1 do
    for j: = 0 to col - 1 do
      p := pixel at row and col in image file
      if p is feature pixel then
        coordinate_X[count_of_feature_pixel] := i
        coordinate_Y[count_of_feature_pixel] := j
        count_of_feature_pixel++;
  for   i := 1 to number_of_sub_image_in_col * number_of_sub-image_in_row do
    send coordinate_X array to No. i slave
  for   i := 1 to number_of_sub_image_in_col * number_of_sub-image_in_row do
    send coordinate_Y array to No. i slave
  for k := 0 to number_of_sub_image_in_row - 1 do
    for l := 0 to number_of_sub_image_in_col - 1 do
      count++
      read each sub-image and set it with No. count
      send No. count sub-image to No. count slave
  count := 0
  for k := 0 to number_of_sub_image_in_row - 1 do
    for l := 0 to number_of_sub_image_in_col - 1 do
      count++
      receive No. count sub-image from  No. count slave
      put this sub-image into the whole image
      get finial result
else then
  receive coordinate_X array from master
  receive coordinate_Y array from master
  for i := 0 to sub_image_volume - 1 do
    for j := 0 to sub_image_volume - 1 do
      if i == 0 || j == 0 || i == subvolume-1 || j == subvolume-1
```

find shortest distance from point (i, j) to feature pixel and record not only this (these) feature pixel(s) but also its (their) coordinates
receive sub-image from master
get distance matrix of distance of this sub-image
send distance matrix to master

3.7 Algorithm analysis

If x>>k2, on the ground that each slave process run on a peculiar core, the worst time complexity of the proposed parallel algorithm is

$$O((k^2 + 4 \times K - 4) \times k^2) \quad (1)$$

In such a scenario, cells in a sub-image are all feature ones and each cell in edges has a different outside nearest feature cell. It can be seen that the time complexity of this algorithm do not relate to the scale of image. Provided that the parallel environment meets requirement, this algorithm can used to compute distance matrix of any scale image with the same time complexity.

4. Conclusion

To meet the demand of processing large size images, a parallel algorithm for calculating cost distance of raster data is proposed in this paper. This algorithm adopts some measures of traditional ones. The analysis shows that it can be used to solve large scale problem.

5. Acknowledgements

The project was supported by the Fundamental Research Founds for National University, China University of Geosciences (Wuhan) under grant CUGL110228 and supported by the high-performance computing platform of China University of Geosciences.

6. References

Dean J, Ghemawat S, 2008, MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1).

Eastman JR, 1989, Pushbroom Algorithms for Calculating Distances in Raster Grids. In: *Proceedings of Amer Soc Photogrammetry & Remote Sensing*, 288-297.

Kang C, 2011, Cloud Computing and Its Applications, Clark university, USA.

Lin C, Snyder L, 2009, *Principles of Parallel Programming*. Pearson Education, London, UK.

Olsson L, 1985, An Integrated Study of Desertification. *Lund Studies in Geography, Ser. C*, (13).

Tomlin CD, Lakey JS, 1983, Three Cartographic Distance-Weight Interpolation Techniques, In: *Proceedings of First Latin American Conference on Computers in Geography.*

Tomlin CD, 1986, The IBM Personal Computer Version of the Map Analysis Package. The Laboratory for Computer Graphics and Spatial Analysis, Harvard University, Cambridge, MA, USA.

Wang K, Han JZ, Tu BB, Dai J, Zhou W, 2010, Accelerating Spatial Data Processing with Mapreduce. In: *Proceedings of 16th International Conference on Parallel and Distributed Systems*, 229-236.