Parallel Geospatial Raster Processing by Geospatial Data Abstraction Library (GDAL) — Applicability and Defects

Li-Jun ZHAN[1,2], Cheng-Zhi QIN[1,*]

[1,*]State Key Laboratory of Resources and Environmental Information System, Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, 11A Datun Road, Anwai, Beijing 100101, China
Telephone: (86-10-64889777)
Fax: (86-10-64889630)
Email: qincz@lreis.ac.cn

[2]Graduate School of the Chinese Academy of Sciences, Beijing 100049, China
Telephone: (86-10-64889461)
Fax: (86-10-64889630)
Email: zhanlj@lreis.ac.cn

1. Introduction

In order to deal with increasingly massive geospatial raster dataset, many researches on parallel geospatial raster processing have been conducted (e.g., Guan and Clarke 2010; Qin and Zhan 2012; Maulik and Sarkar 2012). However, two challenges broadly exist in the input/output (I/O) part of parallel geospatial raster processing, which restrict its application. The first is massive raster data whose movement between fast main memory and slow disk, rather than the computation or communication, often becomes the performance bottleneck of parallel geospatial raster processing. The second challenge is diversity of geospatial raster data formats. Presently there are dozens of often-used file formats for storing raster data. However, existing parallel geospatial raster processing programs often support only very few file formats, which limit the practical applicability.

Some researches have been proposed to address the challenge of massive raster data by providing parallel API to access single file storing massive raster data in specific format, e.g. hierarchical data format v5 (HDF5) and parallel network common data format (PnetCDF), and I/O libraries such as parallel input-output system (PIOS) (Shook and Wang 2011). These researches mainly focus on the I/O performance but pay little attention to the challenge of diversity of geospatial raster data formats.

A widely-used approach to addressing the diversity of geospatial raster data formats in serial raster processing application is to use open-source geospatial data abstraction library (GDAL, http://www.gdal.org/) (Warmerdam 2008). GDAL (with current version of 1.9.2) presents a single abstract data model to read and write a variety of spatial raster data formats. However, there is few literature presenting detailed analysis on the applicability of GDAL for parallel raster processing. In this abstract, two possible I/O modes (serial and parallel) of using GDAL for parallel geospatial raster processing are explored and compared from aspects of efficiency and flexibility.

2. Two I/O modes of using GDAL for parallel geospatial raster processing

Currently most algorithms of parallel geospatial raster processing are based on domain decomposition strategy which the domain processed will be decomposed into subdomains. Thus I/O step in parallel raster processing is to load data of subdomain(s) stored in external memory into internal memory for each process, and later to write the computational result data of the subdomain(s) from internal memory to external memory for each process. There are two possible I/O modes of using GDAL to implement the I/O step of parallel geospatial raster processing.

2.1 Serial I/O mode

Serial I/O mode of using GDAL to assess raster data for parallel geospatial raster processing is illustrated in Figure 1. By this mode all data will be loaded from external memory just through a master process. Other work processes will access the data by communicating with master process. This mode might confront single bottleneck when the raster file is so large that the memory capacity of single compute node will be exceeded.

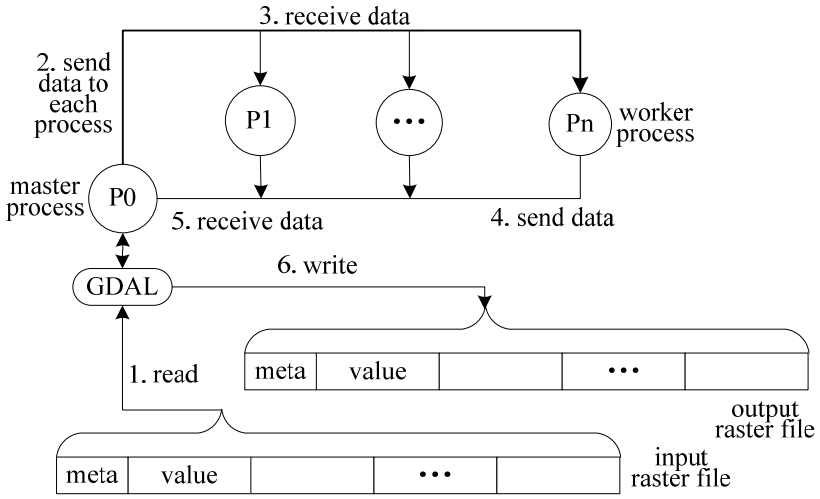


Figure 1. Serial I/O mode of using GDAL for parallel geospatial raster processing

2.2 Parallel I/O mode

Different with the serial I/O mode, parallel I/O mode of using GDAL permits each process to directly read and write data of subdomain(s) stored in external memory (Figure 2). With the parallel I/O mode, the master process uses GDAL to extract the metadata (e.g. spatial extent, projection) from a raster file and correspondingly create an empty output file. Then according to a specific domain decomposition strategy the master process sends the spatial extent information of each subdomain to the corresponding work process. Based on the subdomain information received, each process uses GDAL to read the data of the subdomain. After computing, each process uses GDAL to open the shared output raster file and to write the result data in it. Thus not only the single

bottleneck problem but also the overheads of data distribution between the master process and work processes in serial I/O mode can be avoided.

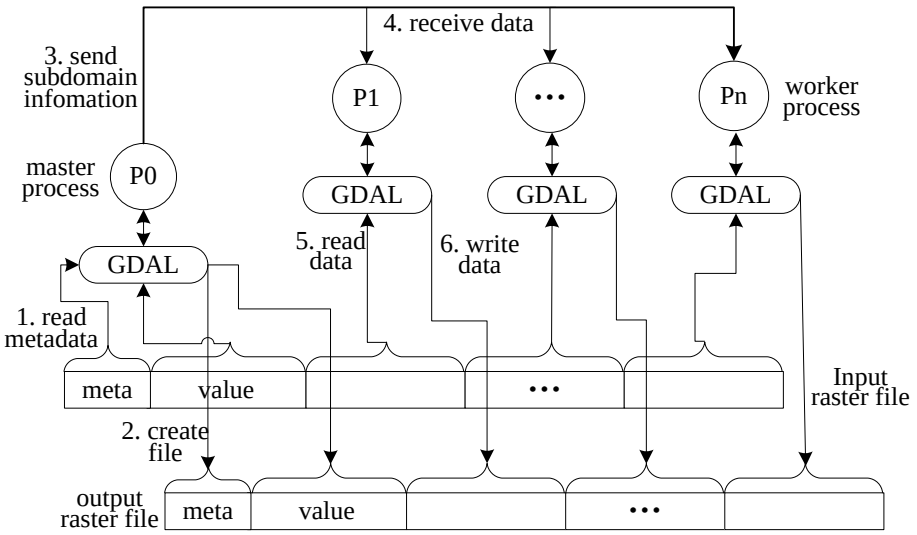


Figure 2. Parallel I/O mode of using GDAL for parallel geospatial raster processing

3. Implementation

Based on message-passing-interface (MPI) programming model, we implemented both serial and parallel I/O modes of using GDAL (called *GDAL_SIO* and *GDAL_PIO*, respectively). Each mode was implemented as that GDAL should support three straightforward domain decomposition strategies which are often used, i.e. row-wise, column-wise, and block-wise (Figure 3).

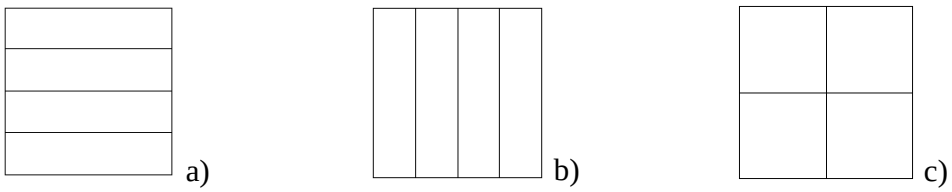


Figure 3. Domain decomposition strategies: a) row-wise; b) column-wise; c) block-wise.

4. Experimental design

The experiments were designed to evaluate the efficiency and flexibility of *GDAL_SIO* and *GDAL_PIO*. Here the efficiency means how quick a mode tested can read and write data in a raster file with specific format. Therefore runtimes of *GDAL_SIO* and *GDAL_PIO* were measured under the same conditions by using the row-wise decomposition strategy for a raster dataset saved as “gtiff” file format. Here the runtime of *GDAL_SIO* includes not only the time for reading and writing data between internal memory and external memory, but also the time for transferring data between the master process and the work processes. The test data is a raster with a dimension of 24496×17100 cells. Both *GDAL_SIO* and *GDAL_PIO* were tested on two hardware environments:

(1) a symmetric multiprocessing (SMP) with two Intel(R) Quad Core E5645 Xeon(R) CPUs (twelve processors) and 32 GB RAM;
(2) a cluster composed of five nodes (one I/O node, four compute nodes). Each node consists of two Intel(R) Quad Core E5645 Xeon(R) CPUs (twelve processors) with 32 GB RAM. Compute nodes share the disk of I/O node through network file system (NFS).

To evaluate the flexibility of each mode, i.e. if the mode works well under different domain decomposition strategies for different raster file formats, *GDAL_SIO* and *GDAL_PIO* using three decomposition strategies (i.e. row-wise, column-wise, and block-wise) for two raster file formats (i.e. "gtiff" and "img") were tested.

5. Experimental results

5.1 Efficiency

The experimental results show that on both SMP and cluster the runtimes from *GDAL_SIO* are much longer than those from *GDAL_PIO* (Figure 4).

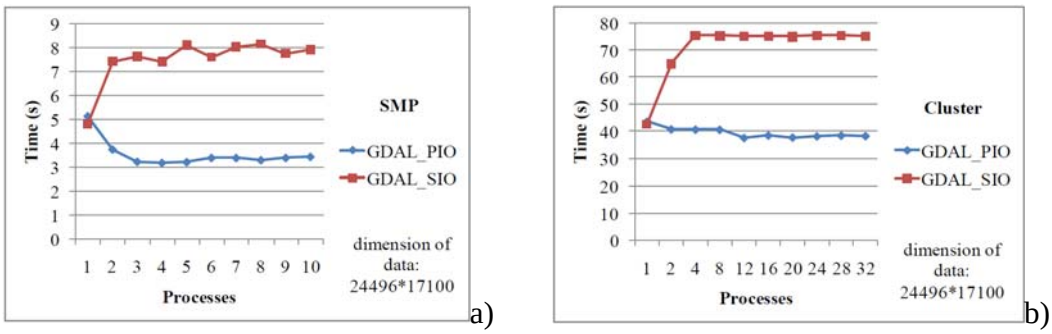
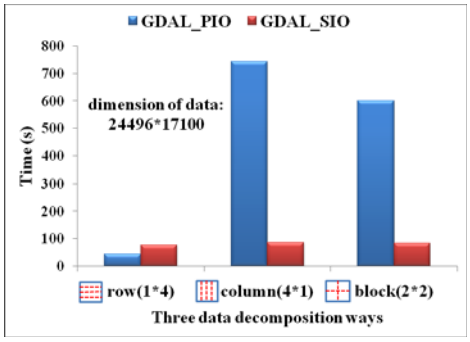


Figure 4. Runtimes from two I/O modes with row-wise decomposition strategy under different numbers of processes (test data is in "gtiff" format): a) a 12-processor SMP, b) a 60-processor cluster.

5.2 Flexibility

The experimental results show that *GDAL_PIO* lacks flexibility, comparing with *GDAL_SIO*. For test data with "gtiff" format, *GDAL_PIO* with column-wise and block-wise decomposition strategies performed highly inefficient, which the runtimes are almost 13~15 times of that from *GDAL_PIO* with row-wise decomposition strategy (Figure 5a). Worse still, *GDAL_PIO* with column-wise and block-wise decomposition strategies got incorrect results which unreasonably contain zones without value (Figure 5b). For the test raster data stored as "img" format, *GDAL_PIO* similarly got incorrect results, no matter which domain decomposition strategy was used. On the contrary, *GDAL_SIO* can work correctly and is adaptable to all three decomposition strategies (Figure 5a).



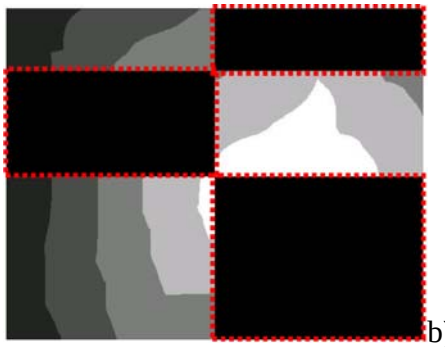
a)

b)

Figure 5. Performance of *GDAL_PIO* executed on cluster with a test raster file in "gtiff" format: a) comparison between two I/O modes of using GDAL with three decomposition strategies; b) example of the incorrect results from *GDAL_PIO* with column-wise and block-wise decomposition strategies (black rectangles in figure are zones without result).

6. Summary

This abstract presents two possible I/O modes (serial and parallel) of applying GDAL to parallel geospatial raster processing. The experimental results show that parallel I/O mode is more efficient than serial I/O mode of using GDAL. However, parallel I/O mode with current version of GDAL is lack of flexibility because it cannot work properly under different domain decomposition strategies for different raster file formats. Now we are fixing this problem in GDAL based on the analysis of the reason for this situation.

7. Acknowledgements

This study was funded by the National High-Tech Research and Development Program of China (2011AA120302) and the Institute of Geographical Sciences and Natural Resources Research, Chinese Academy of Sciences (2011RC203).

8. References

Guan Q and Clarke KC, 2010, A general-purpose parallel raster processing programming library test application using a geographic cellular automata model. *International Journal of Geographical Information Science*, 24 (5): 695-722.

Maulik U and Sarkar A, 2012, Efficient parallel algorithm for pixel classification in remote sensing imagery. *Geoinformatica*, 16(2): 391-407.

Qin C-Z and Zhan L-J, 2012, Parallelizing flow-accumulation calculations on graphics processing units—from iterative DEM preprocessing algorithm to recursive multiple-flow-direction algorithm. *Computers & Geosciences*, 43:1-50.

Shook E and Wang S-W, 2011, A parallel input-output system for resolving spatial data challenges: an agent-based model case study. In: *Proceedings of the ACM SIGSPATIAL Second International Workshop on High Performance and Distributed Geographic Information Systems*, New York, USA, 18-15.

Warmerdam F, 2008, The geospatial data abstraction library. In: Brent H and Michael LG (eds.), *Open Source Approaches in Spatial Data Handling*. Springer, Berlin, pp. 87-104.