A MPI-based parallel pyramid building algorithm for large-scale RS image

Gaojin He, Wei Xiong, Luo Chen , Qiuyun Wu, Ning Jing

College of Electronic and Engineering, National University of Defense Technology, Changsha 410073, China,
Telephone:+8615507486460
Email:gaojinhejs@126.com

Abstract

Building pyramid for remote sensing (RS) image is an effective way to achieve image multi-resolution organization, and also an important way to improve performance of image browsing. For large-scale remote sensing image, traditional sequential pyramid building processing is a time consuming task in many applications. By taking advantage of multi-core, multi-node cluster computing environment and parallel processing mechanism, a MPI-based parallel algorithm is proposed, which can greatly improve the performance of pyramid building. The algorithm has a good scalability and can easily be extended to a considerable number of nodes. Experimental results show that the proposed algorithm has better acceleration effect compared to the sequential methods, and there is a positive correlation between the acceleration effect and image size. For large remote sensing image (in our case 46 GB), the parallel algorithm can be about 10 times faster than GDAL.

Keywords: remote sensing, pyramid, MPI, parallel.

1. Introduction

With the rapid development of remote sensing (RS) image acquisition technology, the spatial resolution and temporal resolution of RS image have been greatly improved, which led to a sharp increase in the size of a signal image (GBs or even TBs). The large size brings a great challenge for image pyramid building. However, traditional serial processing strategy of image pyramid building like GDAL will take quite a long time to process large-scale RS images, how to quickly build pyramid for large-scale RS images becomes an urgent problem. Making use of multi-core, multi-node cluster computing environments and parallel processing mechanism to accelerate the speed of pyramid building is an effective way.

Currently there are two main parallel methods that have made some achievements. One is based on GPU, using the process power of GPU to improve the performance (Chenguang Dai et al. 2011). Another one takes advantage of distributed cluster system, decomposing the pyramid building task into several subtasks which run on different nodes simultaneously (Liu Xiao-li et al. 2014). However, these methods both have their problems. The GPU-based parallel method is hardware-aware. The distributed cluster-based method requires the data to be distributed stored in different nodes, and the complete pyramid file need to be merged, which is time consuming. Therefore using high-performance, disk-shared cluster and MPI (Message Passing Interface) parallel mechanism is an alternative method to build pyramid for large-scale image in parallel. In

this way, the task is decomposed into several subtasks too, but by using MPI/IO, the result of each subtask can be written to the same pyramid file simultaneously.

2. MPI-Based Parallel Algorithm

The parallel algorithm is implemented based on MPI (Message Passing Interface, http://de.wikipedia.org/wiki/Message_Passing_Interface). MPI is widely used in cluster for parallel programming. The main steps of our parallel algorithm are shown as table 1.

The main steps of the parallel algorithm
1. The main process P_0 reads metadata of the RS image, which contain the number of bands, row size, column size, data type. etc.
2. The main process P_0 creates a blank image pyramid file based on the metadata and the pyramid level specified by the user. The format of the blank file is geoTIFF(A TIFF based interchange format for georeferenced raster imagery). The blank file only contains metadata, which just specify the pyramid structure. And if the pyramid file is larger than 4G, the file will be created in bigTIFF.
3. The main process P_0 partitions the RS image into several parts. And then notify other processes to read data from the image. Each process reads a subdomain of the whole image.
For i = 1 to BandNum (the number of bands of the RS Image):
4. The processes (P_0,...,P_{N-1}) read the data in their own domains into memory in parallel.
For j = 1 to PyramidLevel (the level specified by the user):
5. The processes (P_0,...,P_{N-1}) begin data resampling. The degree of resampling is different for different pyramid level.
6. The processes (P_0,...,P_{N-1}) open the blank pyramid file created in step 1 in share-mode and then write the result in the proper position in parallel.

Table 1. The main steps of the parallel algorithm

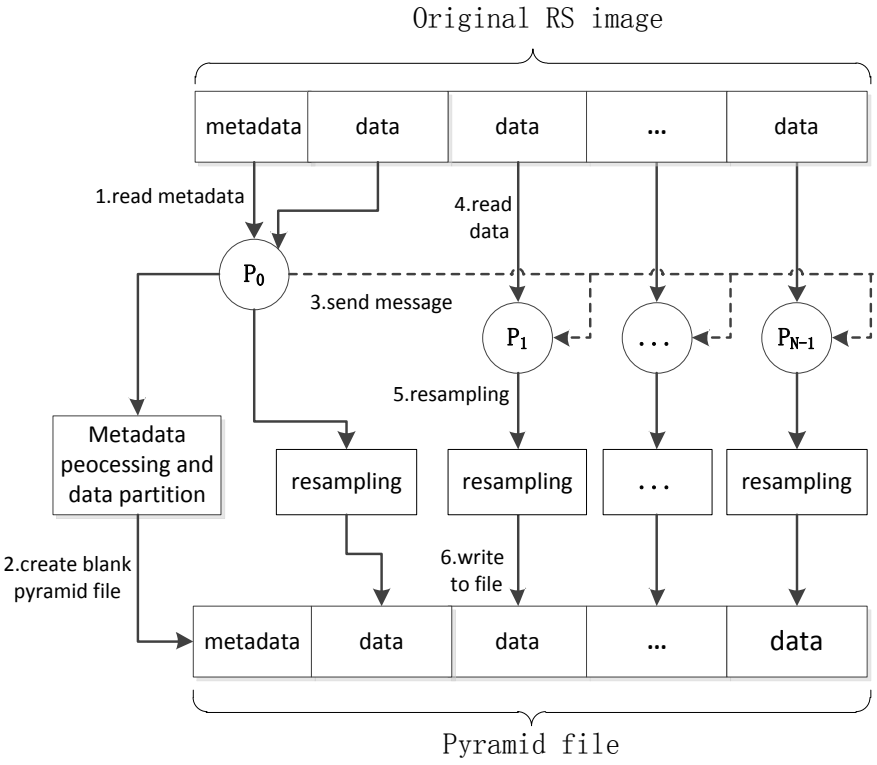

Figure 1. The main steps of parallel building image pyramid

The image pyramid generated by the parallel algorithm has the same format with the pyramid file generated by GDAL, so it can be used by most of the main GIS products directly.

3. Implementation

3.1 Data Partition

There are three common ways of domain decomposition for parallel processing: row-wise, column-wise, and block-wise. The row-wise method has the best performance for geospatial raster data (Qin Cheng - Zhi et al. 2013). In fig 2, we assume that the image size is XSize*YSize, the number of band is K, pyramid level is M, and the number of processes is N. For each band of the image, the size (measured in rows) of the data processed by P_i is defined as equation 1.

$$\text{subRow}(i) = \begin{cases} \lfloor \text{YSize}/N \rfloor, & i < N-1; \\ \text{YSize} - (N-1)\lfloor \text{YSize}/N \rfloor, & i = N-1. \end{cases} ,(i=0,1,\ldots N\text{-}1) \quad (1)$$

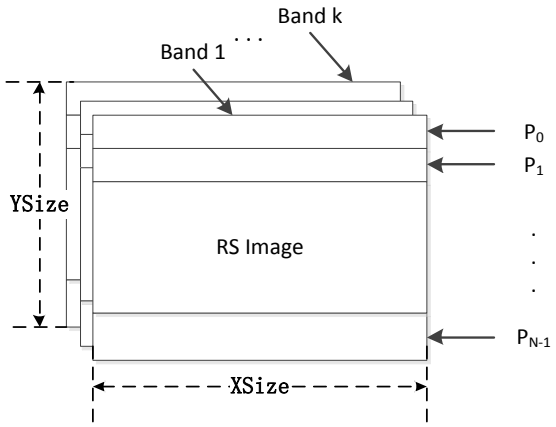

Figure 2. Data partition

And the offset (in rows) of the i-th subdomain that processed by P_i is shown in equation 2.

$$off(i) = i * \lfloor YSize/N \rfloor, (i=0,1,...,N-1) \quad (2)$$

3.2 Parallel Resampling

Resampling is the key step in RS image pyramid building. In general, the common pyramid algorithms (Nearest Neighbour, Bilinear. etc.) all run in sequential mode. In order to take full advantage of multiple processors, we designed a parallel resampling algorithm based on the common pyramid algorithm. The whole image domain is decomposed into N subdomains (SD_0, …, SD_{N-1}), each process does the resampling independently.

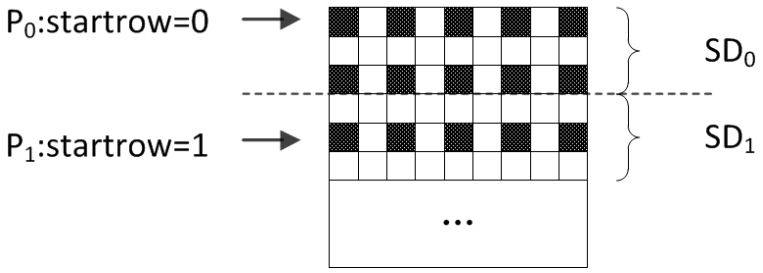

Figure 3. The resampling algorithm is Nearest Neighbour. The squares with shadow represent the sampling pixels. For P_0, the *startrow* is 0, while P_1 is 1.

Since the data processed by one process are only a part of the whole image, processes can't always sample data from the first row in their domains (fig 3).. For P_i, the start row in its domain SD_i can be expressed as equation 3.

$$startrow(i,m) = \begin{cases} 0, & off(i)\%2^m = 0; \\ 2^m - off(i)\%2^m, & off(i)\%2^m \neq 0 \end{cases}, (m = 1, 2, \ldots, M) \quad (3)$$

3.3 Parallel I/O

When building pyramid for large-scale RS image, I/O is a very time-consuming operation taking even longer time than resampling. Therefor increasing I/O bandwidth is of great significance to speed up pyramid building. While parallel I/O is an effective way to improve I/O performance (Ferhatosmanoglu Hakan et al).

GDAL is used for parallel reading. Research shows that using GDAL for parallel reading can effectively improve the performance . And we use the MPI/IO library for parallel writing. MPI/IO is a library providing high performance, portable, parallel I/O interface to high performance MPI programs. MPI/IO can achieve high I/O performance on parallel file system.

4. Experimental Results

The parallel algorithm was tested on an IBM SMP cluster with 32 server nodes. Each node consists of a 16-core CPU (Intel Xeon E5-2640, 2.0 GHz) and 32GB DDR3 memory. The MPI implementation is Intel MPI 4.1 and the parallel file system of the cluster is IBM's GPFS. The images used for testing are list in table 2.

Image name	Dimension	Type	Size
d1	44800*36864	8-bit	1.6 GB
d2	87040*58368	8-bit	4.8 GB
d3	77312*99328	8-bit	7.2 GB
d4	220672*86272	8-bit	18 GB
d5	136448*90368	32-bit	46 GB

Table 2. RS Images used for testing

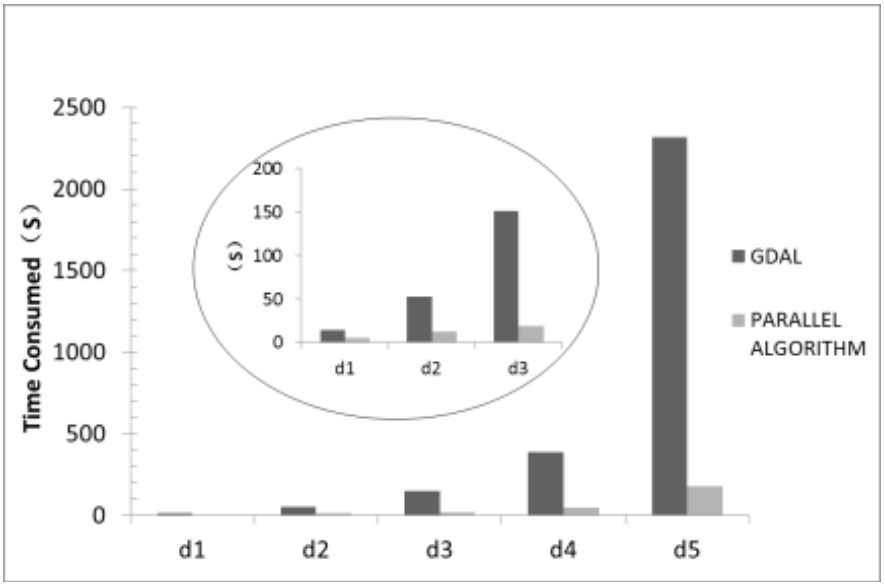

Figure 4. The time consumed for pyramid building with RS images in different sizes. In this experiment, our algorithm uses 16 processes.

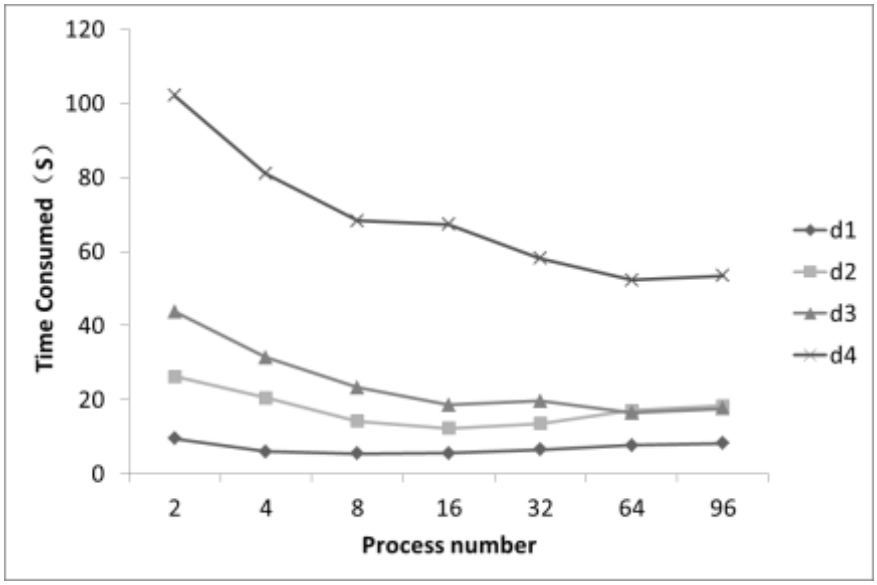

Figure 5. The performance of our parallel algorithm varies with different number of processes. The images used are d1, d2, d3 and d4.

Compare our parallel algorithm with GDAL (fig 4). The figure shows: (1) The parallel algorithm has obvious acceleration effect compared to GDAL with images of various sizes. (2) The acceleration effect becomes more obvious with the increasing of the image size. For RS image in size of 46 GB, our parallel algorithm can be 10 times faster than GDAL.

Algorithm performance varies with different number of processes (fig 5). The figure shows: (1) the algorithm performance improves with the increasing of the number of processes. This indicates that accelerating the pyramid building by increasing the number of processes is feasible. (2) For images in small size, when the process number is large, performance will be a little lower. This is because the cost of maintaining numbers of process and fragmented I/O counteract the benefits of parallelization. So it is unwise to specify too many processes for small size image pyramid building.

5. Conclusion

In this paper, we propose a parallel algorithm for large-scale RS image pyramid building based on MPI. In our algorithm, resampling and I/O which are the two main steps are carried out in parallel. The experimental results demonstrate that the proposed parallel algorithm can significantly improve the performance of the pyramid building for RS image.

6. Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant 41271403, 41471321) and the National High Technology Research and Development Program of China (Grant 2012AA12A405).

7. References

Chenguang Dai, and Yang Jingyu. "Research on Orthorectification of Remote Sensing Images Using GPU-CPU Cooperative Processing." Image and Data Fusion (ISIDF), 2011 International Symposium on. IEEE, 2011.

Kang, Jun-Feng, et al. Parallel image resample algorithm based on GPU for land remote sensing data management. Journal of Zhejiang University (Science Edition) 6 (2011): 018.

LIU Xiao-li, XU Pan-deng, ZHU Guo-bin & LI Xue. Parallel and Distributed Retrieval of Remote Sensing Image Using HBase and MapReduce. Geography and Geo-Information Science, 30(5), 2014.

Bassett, Michael. MPI on the Move[J]. Corporate Meetings and Incentives, 2006, 25(7): 9-10.

Qin, Cheng-Zhi, Li-Jun Zhan, and A. Zhu. How to Apply the Geospatial Data Abstraction Library (GDAL) Properly to Parallel Geospatial Raster I/O?. Transactions in GIS, 2013.

Porwal, Shardha, and Sunil Kumar Katiyar. Performance evaluation of various resampling techniques on IRS imagery. Contemporary Computing (IC3), 2014 Seventh International Conference on. IEEE, 2014.

Ferhatosmanoglu Hakan, Agrawal Divyakant, Egecioglu Omer. Optimal data-space partitioning of spatial data for parallel I/O[J]. Distributed and Parallel Databases, 2005, 17(1):75-101.