A Fast Tile-Pyramid Construction Algorithm Based on Multilevel Task Parallelism

Haozheng Liu1, Ye Wu2*, Ning Jing3, and Shihao Feng4

1,2,3,4College of Electronic Science and Technology, National University of Defense Technology, Changsha, China

*Corresponding author: yewugfkd@nudt.edu.cn

Abstract. With the rapid development of satellite sensor technology and aerial photography technology, spatial and temporal resolution of the remote sensing image has been greatly improved. To speed up the visualization of remote sensing images, pyramid building technology is the best choice at present. However, the performance of existing methods is not as good as expected. Considering the load imbalance problem caused by the uneven data distribution problem, this paper proposes a fast tile-pyramid construction algorithm based on multilevel task parallelism. Our method changed the traditional parallel tile generation algorithm which is generated layer by layer and implemented a new algorithm that generated multiple layers at the same time by using the entire tile pyramid as a whole. Experimental results on real-life datasets show that our new algorithm can improve the construction speed of tile-pyramids.

1. Introduction

With the development of the geographical spatial data acquisition technology, many organizations and institutions can be convenient to get a lot of high-resolution map images [1]. So, the rapid processing of high-resolution map images makes it an urgent need for network geography applications [2]. The traditional large geographical spatial raster data visualization needs to spend a lot of time, including coordinate transform, image matching, image texture and tile pyramid building, etc [3]. Generally speaking, it takes several days to process raster image data with a data volume of hundreds of GB and a resolution of 0.1 M. In terms of timeliness, this visualization speed has been far behind the actual demand. To solve this problem, the researchers developed tile technology [4], which cuts images into a set of pictures of a certain size according to certain mathematical rules and stores them on a server. When accessing the map service through the client, the tiles are not created by the server in real-time and transmitted to the client. Instead, the server directly returns the tiles corresponding to the region and forms the map at the client, thus reducing the burden on the server and improving the speed of map browsing. Nowadays many scholars have conducted many studies on how to generate tiles faster and how to store and access tiles faster, most tiles formation and organization methods have its advantages and disadvantages. The traditional tile generation methods often have some problems, for example, the generation speed is not fast enough, the generated data takes up a lot of space, and the access delay is high. This paper proposes a new method for higher-speed tile generation, the method starts from the principle of tile pyramid generation and implements a new multi-level task parallel method to solve the problem of uneven task load during tile generation. To a certain extent, the speed of tile generation has been improved, and the visualization effect of raster image maps has been optimized.
2. Related work
The generation of map tiles has already been a hotspot technology for GIS at home and abroad. In recent years, many researchers have done a lot of research on how to provide better map tile services in many aspects. In literature[5], N.Guo et al. adopted an adaptive multilevel tiles generation method which first builds a grid index for the geospatial raster dataset, and then generates tiles according to different hierarchy level numbers in the tile-pyramid tile. This method is mainly optimized for the visualization stage of map tiles and improves the visualization efficiency of map tiles for large-scale raster data. In [6], Yu Kai et al. designed a cloud storage system for map tiles, which realized the rapid storage and efficient query of massive tile data through the MapReduce parallel computing framework and the Hilbert space-filling curve. This system mainly started from the storage structure to improve the tile service and effectively improves the management efficiency of massive tile data. To solve the problem that the existing commercial geographic information system platform is difficult to flexibly integrate the use of multi-source image data fusion, Fan J F et al.[7] studied the spatial reference differences and tile organization rules of multiple data sources under the Cesium platform. According to the mashup design of the tile data sources with different spatial references, a static and dynamic loading scheme of multi-source local tile data based on Cesium is proposed. This solution has made further research on the data source and loading method of tiles to improve the quality of map tile service. The above studies have optimized the storage, index, data source, and loading method of map tiles to a certain extent. However, few people have improved the algorithm from the generation principle of map tiles. For the rapid generation of map tiles, the improvement in the principle of tile generation has a wider application value. Liu et al. proposed a method for directly generating low-level tiles from a high-level level[8]. This method improves the original generation principle of traditional map tiles. When building a tile pyramid, it is no longer to generate an image pyramid for each layer and then cut to generate a tile pyramid, but only to generate an image pyramid for the highest layer and then use resampling to generate other levels of map tiles. This method greatly improves the efficiency of tile pyramid generation and effectively reduces the time required for the visualization of map tiles. However, as the former mentioned methods generate tiles by a layer-by-layer approach, it is easy to lead to load imbalance during parallelization. To solve this problem, this paper proposes a new multi-level task parallel map tile generation algorithm.

3. Multilevel task parallelism

3.1. Use the entire pyramid as a task division unit
The traditional tile pyramid construction algorithm takes each layer of the tile pyramid as the task division unit, first calculates the number of tiles in each layer as the total number of tasks, and then divide all tasks among each process based on the number of processes [9]. This method first requires task division for each layer of the tile pyramid, and starts a process pool at the same time. When the tasks of each layer are completed, the process pool needs to be closed, and the process pool is restarted when the next layer of tasks starts. As the total number of tile pyramid levels increases, the overhead of the creation process is increased. At the same time, due to the uneven distribution of data, there is a problem that the process completed first needs to wait for the process completed after the completion of the task before entering the next task. The new algorithm proposed in this paper uses the entire tile pyramid as a task division unit, and perform task division only once for the construction of a tile pyramid. The new algorithm first calculates the number of tiles in each layer, accumulates the number of all tiles, and uses the total number of tiles as the total number of tasks. Each process is evenly distributed according to the number of processes. After the task is started, the process pool only needs to be started once, and the process pool is not closed until the entire pyramid is completed. It reduces the overhead caused by unnecessary creation processes. At the same time, the process that finishes tasks first can go directly to the next layer for tasks without waiting for all tasks in this layer to complete. This method has greatly reduced waiting time and improved construction speed of the entire
tile pyramid. However, there is a task dependency when process generate higher level tile in the part of resampling.

3.2. Solution of task waiting when resampling

In fact, the tile generation method using resampling must have the problem of task dependency which is shown in the following figure 1. In the process of tile generation, there is a problem that each tile task spends different time because the raster data is not evenly distributed or the process are load imbalance. So, the task which completed first needs to wait for the other tasks particularly in the part of resampling. The task dependency results in a lot of time wasted in generating the pyramid of tiles because of the process waiting for each other. In order to solve the problem of task dependence, the traditional tile pyramid uses a layer-by-layer parallel algorithm, mainly because low-level tiles are generated by resampling [5]. If you want to generate a tile of layer K (K is not the highest level of tiles), the corresponding four tiles of the layer K + 1 need to be generated first, and then the tile of layer K will be generated by resampling. Therefore, a layer-by-layer parallel algorithm can guarantee that when the K layer tiles need to be generated, the (K + 1)-layer tiles have completed the task. The multi-level parallel algorithm in this paper needs to solve this problem first to ensure the completeness of the construction of the entire tile pyramid. The method used in this paper is to make a judgment before each tile task is started. When the corresponding four tiles of the (K + 1) (K is not the highest number of tiles) layer have been generated, the tile task proceeds normally. When the condition is not met, the process maintains a wait state until the condition is met. At the same time, in order to reduce the overhead of repeatedly reading the file, this article uses a built-in global list method to complete this judgment module. For each tile, the corresponding serial number is calculated by the task division method. When the tile task is completed, the serial numbers of the tiles are added to the list, and the judgment during resampling only needs to retrieve the four serial numbers of the corresponding tiles in the list to complete the judgment.

3.3. The detailed steps for each process

Based on the above two points, each process in the parallelized tile algorithm completes the task according to the following step. First, the tasks are divided equally according to the total number of tile tasks and the total number of processes, and each process obtains tasks in order according to their corresponding process numbers. Therefore, when the task is assigned, the process will obtain the corresponding serial number of the tile of the current task. This part is completed by an iterative counter, through this counter, each corresponding process will be arranged in order from the highest-level tile to the lowest-level tile. For each process itself, the order of completing tasks is also from high to low. After determining which tile task to execute, the process will execute the tile generation task and when the task ends, the process will write the i (the serial number of task) to the list, indicating that the tile has been generated. However, since the highest-level tiles generation are different from other-level tiles which generating tile directly from remote sensing image but lower-level tiles use the tile generation method of resampling. So, for each tile task, it is also necessary to determine whether the tile should perform the highest-level tile directly generating task (base tile generation) or the low-level tile resampling generating task (over tile generation). Because the experimental code in this article used the ‘multiprocessing’ module in Python, the tile generation uses the ‘Z / X / Y’ file system organization method, so the number of tiles in each layer can be obtained directly by the formula (1).
Fig. 1. The task dependency in the process of tile pyramid construction

From this, it can be judged whether the process needs to do base tile generation or over tile generation., in other words, whether the current counter \( i \) is greater than \( N_0 \) (the highest number of tiles). Moreover, in next step of the process, there is a judgment when doing the over tile generation. If this over tile generation task meets the condition of the judgment, then process will do the tile resampling work, otherwise the process does an endless loop until the condition is met. This condition is whether the four groups of tiles at the previous level which corresponding to the current tile task has been generated as described in III.B. To achieve this judgment, we only need to first calculate the serial numbers of the four tiles corresponding to the previous level, and then retrieve whether the four sequence numbers are in the list. The calculation formula of the four tile serial numbers \( I_0 \) corresponding to the current tile is as follows.

\[
for \ y \ in \ range(2*ty,2*ty+2) : \\
x \ in \ range(2*tx,2*tx+2) :
\]

\[
N_{alt} = N_0 + N_1 + \cdots + N_{k-1} \\
I_n = N_{alt} + (Y_{max} - y) \times (1 + X_{max} - X_{min}) \times (1 + x) \\
\text{for} \ n = 0,1,2,3 \quad (2)
\]

The \( tx \) and \( ty \) are the serial numbers of \( x \) and \( y \) corresponding to the current tile. \( N_{alt} \) represents the total number of tiles from 0 to \( K-1 \) above the \( K \)-layer pyramid, \( X_{max} \) and \( Y_{max} \) represent the serial numbers of maximum \( X \) and \( Y \) of the current \( K \)-layer pyramid. By knowing the serial numbers of \( x \) and \( y \) corresponding to the current tile task, the serial numbers of \( x \) and \( y \) corresponding to the four tiles in the previous layer are obtained. Then through calculation, the serial numbers \( I_n \) (\( n = 0,1,2,3 \)) corresponding to the four tiles can be obtained. Finally, the judgment stage can be completed by retrieving whether the serial number of the corresponding tile exists in the list. If the serial number exists in the list, it performs the next tile task, and if the serial number does not exist, it does a circular wait until the condition is met. The waiting of this process is different from the waiting of the entire level in the traditional algorithm. There is only a short wait for a single tile task, and the entire pyramid generation is still carried out simultaneously at multiple levels. In the actual generation process, the waiting of the process is very necessary, and it is beneficial to the generation speed of the entire pyramid generation. The pseudocode of the entire process is as follows.
4. Experiments and results

We implement the new tile-pyramid building method based on a high-performance GIS platform named HiGIS. The performance of this method is compared with the paratileM [8]. We only compare the resampling part of all the algorithms.

The experiments are conducted on a TIFF raster data with size of 6.7GB.

4.1. Cost comparison

We carried out experiments on the number of processes of 4, 8, 12, 16, 20, 24, 28, 32 under the 6-level tile pyramid. The time costs compared with paratileM algorithm are shown in Figure 2.
The experimental results show that the new algorithm is faster than the old one among all the experimental results. Because the new algorithm can effectively reduce the waiting time of the process between the pyramid layer and the layer, it is obvious that the new algorithm will take less time than the original algorithm in the experimental results. As the number of processes increases, both algorithms have not been significantly improved. From this point of view, the new algorithm just improved the problem of load imbalance to some extent.

4.2. The degree of parallelism of the program

In this experiment, we test the degree of parallelism of the algorithm by running the program under the 6-level pyramid and then take the average of the 5 test results as the final experiment result which is shown in the Figure 3.

Figure 3 shows the transformation of the total algorithm time with the total number of processes. The abscissa is the number of processes and the ordinate is the algorithm time. From the experiment results, it can be seen that the algorithm speed increases gradually as the number of processes increases. After a certain degree, the tile generation speed gradually approaches a stable value and does not continue to increase. The main reason is that when the number of processes gradually increases, the hard disk read and write speed reaches the upper limit, and the algorithm speed cannot continue to increase. Figure 4 shows the speedup of the algorithm, the abscissa is the number of processes and the ordinate is the acceleration ratio. It can be seen from the figure that when the number of processes reaches 16, the increase rate of the acceleration ratio starts to decline. It has a roughly linear relationship with the number of processes. From Figure 4 we can see that the degree of parallelization of the algorithm is still high.
5. Conclusion and future work

This paper proposed a new type of tile generation algorithm, which is improved from the principle of the pyramid construction algorithm. The experiment results verified that the algorithm has improved the load imbalance problem in the resampling part of the pyramid construction. It is also capable of supporting the full part of the pyramid construction and larger data test. However, due to the lack of data and experimental tools, this article did not achieve an experimental comparison between higher-level pyramids and larger amounts of data. It is expected that it will be further supplemented in future experiments to achieve all parts of the pyramid construction and experimental analysis of larger amounts of data.

Acknowledgments

This research was supported by the Natural Science Foundation of Hunan Province under Project (2019JJ50718) and National Natural Science Foundation of China under Project (4187284)

References

[1] M. Ömer, A. Ceylan, J. Geogr. Reg. Plann, 2, 235(2009).
[2] M. J, D.S. Guo, COMPUT ENVIRON URBAN, 33, 403-408(2009).
[3] X. Zhang, M. Pazar, CaGIS, 31, 29-41(2004).
[4] F. Yin, L. Sun, Geomatics & Spatial Information Technology, 5 (2010).
[5] N. Guo, W. Xiong, Q. Wu, N. Jing, AECE, 16, 3-8(2016).
[6] J. Fan, T. Hu, H. He, L. Qin, G.H. Li, JARS, 23, 695-705(2019).
[7] W. Ma, J. Ren, Q. Zhu, Journal of Geomatics 2018, 43, 66-69(2018).
[8] S. Liu, Q. Wu, L. Chen, J. Li, N.Jing, Geomatics World, 11(2015).
[9] S. Liu, Q. Wu, L. Chen, J. Li, N.Jing, ComEngApp, 7(2018).