Parallel LOD building for large scale terrain visualization

Lei Liu, Yong Gao*, Hao Yu, Xiao Guo

Institute of Remote Sensing and Geographical Information System, Peking University, Beijing, China

Telephone: 86-010-62751186

Email: gaoyong@pku.edu.cn

1. Introduction

3D visualization of large scale terrain data brings up a lot of problems. The main problem in real-time graphics is rendering efficiency. To best exploit the rendering performance, the scene complexity must be reduced as much as possible without leading to an inferior visual representation. Since the massive terrain data cannot be loaded into memory once, Level-of-Detail (LOD) technology and multi-resolution pyramid are used. There are many works done about out-of-core loading (Lindstrom, 2002; Danovaro, Floriani, etc, 2005) and rendering the pre-build terrain LOD data in a cluster or parallel environment (Goswami, Makhinya, etc, 2010; Yin & Shi, 2005) to accelerate the rendering speed. In fact, it's quite time consuming to generate the terrain LOD data but little have been done about how to build them quickly. In this paper we propose an efficient way for generating terrain LODs in a cluster environment. In order to decompose the task, we use quadtree, in which each node can be a separate task for its space area, to organize the terrain data.

2. Terrain LODs structure and storage

To represent terrain data in a multi-resolution model, an efficient hierarchical spatial data structure is important. For large scale terrain data, quadtree based multi-resolution triangulations have been shown to have the highest performance in terms of triangulation speed, representation cost, and storage and retrieval efficiency (Pajarola, 2002). So we choose and implement a quadtree to organize terrain data.

Since the terrain data is too large to be loaded into main memory once, it has to be dynamically loaded on-demand from disk. We maintain it on disk partitioned into files of 65x65 vertices each. The number of vertices we choose is tested, and we think that vertices number 65 is balance for disk-loading and file size. All levels of the quadtree hierarchy are stored on disk, each as a set of blocks of 65^2 vertices. We use a scene graph model to organize and render these LOD data. A scene graph model is a hierarchical structure by which an entire tree-representing the virtual world-is organized for efficiency and easy management (Ames, 2002). The tree structure of the scene graph model is fit for out-of-core LOD-based terrain data loading and rendering, which is a similar hierarchical structure too.

The appearance of cracks between neighboring tiles with different levels is a common problem in the multi-resolution model. There are several ways to fix the cracks. Since that in our parallel tasks the tiles in the divided level knowing nothing

about their upper levels, the methods those changing the connectivity of the vertices at tiles' edges are not applicable. The "skirt" method (Ulrich, 2002) that not changing the tiles themselves is just fit for our application. The idea is to simply create a vertical "skirt" around the perimeter of each tile. This method gives a slight increase in polygon count. However, compared to the cost of CPU side stitching by other methods, it is virtually free. At the same time, it is much more simple to implement.

3. Parallel Terrain LODs building

The process of building LODs can be easily paralleled when data are organized with the quadtree structure. The task can be decomposed as Fig. 2.

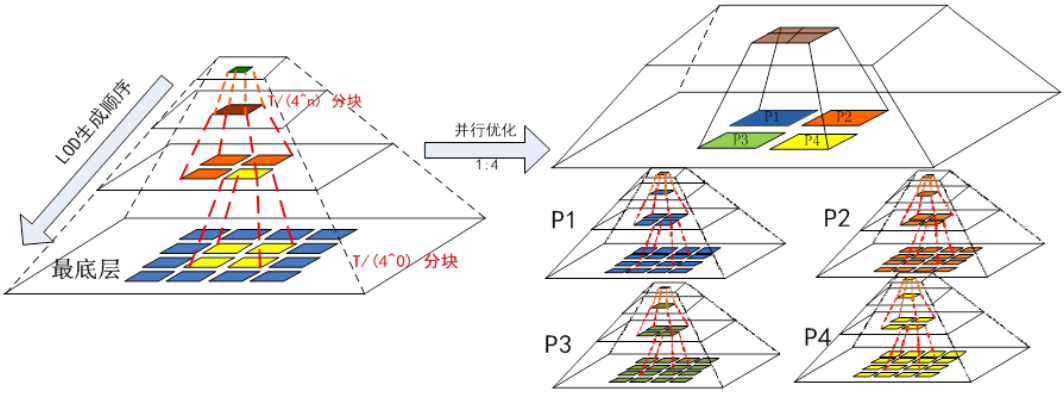

Fig.2 building Terrain LODs in parallel

Starting from the root level (the lowest resolution level, recorded as Level 0), the task is decomposed into four sub tasks by the region and the next level can decomposed the four tasks recursively until the max resolution level or the max level set by the program is reached. Generally the resolution of the next LOD level is twice of the current level in the multi-resolution pyramid. Then the max resolution level *m* will be as equation 1:

$$m = \max\{ floor(\log_2 \frac{w}{2^n}),\ floor(\log_2 \frac{h}{2^n})\} \qquad (1)$$

where n is the point number of every terrain tile, w is the row number and h is the column number of source terrain data. Here in the paper n is set to 65, which we think is reasonable for dynamically out-of-core file loading. Sometimes as we do not need so many LOD levels and want to set the levels manually, the actual max level *l* will be decided by equation 2:

$$l = \begin{cases} n, & n \le m \\ m, & n > m \end{cases} \qquad (2)$$

where n is the level we set manually.

It's quite awful if every tile of the LODs is dispatched as a task, because the task number will be huge and each will load the sources across the network of the cluster which is quite slow. So it's a good idea to make just some tasks to be paralleled and each to do several LOD levels. We decompose the task into two or three units by virtue of the max level as equation 3:

$$if\ \ l<10, n=l/2$$
$$else\ n_1=l/3, n_2=(l\times 2)/3, if\ \ l\ \geq 10 \qquad (3)$$

If the max level is 10, for example, then the first task unit just builds the LODs from level 0 to level 2. The second unit has more tasks, building LODs from level 3 to level 5. The last unit has most tasks, with each building LODs from level 6 to level 10 in its corresponding area.

When the raw task is decomposed in this way, the number of parallel tasks is modest. In this case, it is about 500. The dispatch of tasks as this way can accelerate the LOD building process much which we will describe in section 4.

4. Implement and Results

We implement a preprocess system for building terrain LODs in parallel in a Linux cluster environment. The terrain data is stored in the server. The server is the task manager that provides the ability to do parallel builds, and works out how to decompose the build into small tasks, and then dispatches these tasks through SSH protocol to clients across the network. The actual building process is executed in clients. Clients read data from the server and start tasks in the local machine. Then the LOD files generated by the clients are sent back to the result directory in the server.

Three experiments were taken by different numbers of client nodes. The DEM data is 10GB, the texture is 50GB. Results are shown in Table 1.

Experiment No.	Client Number	Time Cost(hours)	Accelerating rate
1	1	35	1.0
2	2	21	1.7
3	3	15	2.3

Table 1. Accelerating rates of LOD building in a cluster environment

The results shown that with more client nodes, the time cost is becoming less effectively. And with our rendering system implement by a scene graph model, its rendering speed is acceptable, which is always above 20fps.

5. Conclusion

In this paper a new method on building terrain LOD in a cluster environment has been presented. We have shown that the parallel building method is time saving and can be simply implemented. And with a scene graph model, even a single PC can render the preprocessed large scale terrain data. It will be quite helpful in geographic information system or some other terrain visualization system.

6. Acknowledgement

This research was supported by The National High Technology Research and Development Program of China (Grant No. 2011AA120303).

7. References

Pajarola,R., 2002. Overview of Quadtree-based Terrain Triangulation and Visualization (Technical Report UCI-ICS-02-01). *Information & Computer Science*, University of California Irvine.

Ulrich,T., 2002. Rendering massive terrains using chunked level of detail control. *Course Notes of ACM SIGGRAPH.*

Lindstrom,P., Pascucci,V., 2002. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *Visualization and Computer Graphics, IEEE Transactions*: 239-254.

Danovaro, E., Floriani, L., Puppo, E., etc, 2005. Multi-resolution out-of-core modeling of terrain and geological data. *Symposium on Advances in Geographic Information Systems*, New York, ACM Press: 143-152.

Ping Yin, Jiaoying Shi, 2005. Cluster based real-time rendering system for large terrain dataset. *Ninth International Conference on Computer Aided Design and Computer Graphics*: 365-370.

Ames, W., 2002. Real-time terrain rendering and scene graph management. *Notes at Computer Science Department, Boston College*.