An Efficient Mechanism for Organizing and Indexing Tile Caches on Map Server

Haiting Li[1,2]
1 School of Resource and Environment Science
Wuhan University
Wuhan 430079, China
Haiting_cn@163.com
2 Wuhan Geotechnical Engineering and Surveying Institute
Wuhan 430022, China

Lifan Fei, Huibing Wang, Yanhong Li
School of Resource and Environment Science Wuhan
Wuhan University
Wuhan 430079, China
vrwuhan@qq.com

***Abstract*—With the increasing popularity of global on-line mapping web applications (e.g. Google Maps, Microsoft Virtual Earth, Yahoo Maps), Tile caches technique is widely used. Tile caches are a set of map images that have been pre-generated based on geographic vector data and the adoption of tile caches technique in the developed system dramatically decreases the loading time consumed for high quality image visualization. This paper proposes an efficient mechanism for organizing and indexing tile caches on map server. We cached all the map tiles using the same matrix notion to give each tile a unique file name and all tiles are stored in hierarchical file system directories using the zoom level and layer as the directory name. With the number of tile files growing exponentially along with the increase of magnification level and operating system limitations for numbers of files in a single directory, we propose an efficient grid-index algorithm to re-organize the files and directories. This paper gives the implementation of this mechanism in Wuhan City, China. Finally, this paper analyses the shape deformation in specific geographical regions which caused by this algorithm and presents two ways to solve the problem by modifying the tile's width-to-length ratio or changing the step-length of longitude and latitude that each tile covers.**

Keywords: tile caches; organizing mechanism; grid-index algorithm; matrix; deformation; spatial reference system; step-length

I. INTRODUCTION

Response speed has once been the bottleneck of web mapping applications on the internet. With the increasing popularity of global on-line web mapping applications (e.g. Google Maps[1], Microsoft Virtual Earth[2], Yahoo Maps), tile caches technique is widely used. With this technique, the performance had been improved when the users zoom or scroll the map. In essence, the tile caches are pre-rendered and pre-generated [3] by special program based on geographic vector data. All the tiles are organized and stored on the map server in the way of different hierarchies. When the client posts the special request for map, the web applications would send the corresponding map tiles to the client side. With the pre-determined mechanism on the server, the applications need not to draw the map in real time as traditional WebGIS do. So, the tile caches technique is much better in the displaying speed than before. How to pre-generate the tile caches and send the special map tiles according to the client's request? It is necessary to build an effective organizing and indexing mechanism. On the developing process of www.vrwuhan.com website, we propose an efficient algorithm to index all the tile caches based on grids.

II. REPRESENTATION OF TILE CACHES

A tile caches service has a set of map images that have been pre-rendered for rapid display [4]. Tile caches are those map images that have been pre-generated based on geographic vector data. When the client sends a special map request, the corresponding map tiles would be retrieved without requirement of any database, which would decrease the response time of the server. In designated areas, the pre-rendered and symbolized map will be split in vertical and horizontal orientation separately at pre-determined scale levels into many tiles. All the tiles are in fixed size and file format (e.g. Jpeg, png, gif etc) and these rectangular tiles are called map tiles or tile caches. The tile caches are named with different rows and columns in the same zoom level according to their own special location. With different hierarchies tile caches form a similar T-Pyramid structure (Fig.1). Tiles in an index are stored using this structure to allow fast access and scroll operations. Some researchers presented quad tree structure to index the tiles on the caches server [5]. This paper will describe the specific mechanism and algorithm in detail in follow sections.

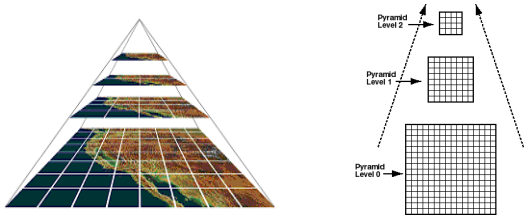


Figure 1. Representation of tiles in pyramid [5] and Pyramid levels [6]

Funded by Wuhan construction research program (200724)

III. EFFICIENT ORGANIZING AND INDEXING MECHANISM

A. Basic Agreement

We suppose the map would be split into n zoom levels. The whole map adopts WGS-84 as the spatial reference system which is located using latitude and longitude. We also assume that each tile represent equal latitude and longitude value at the same zoom level and the difference value of latitude or longitude each tile covers is called step. For the convenience of computing, we adopt non-geometric sequence as ratio of former and latter in longitude and latitude's step array (*Zoomlevel, Longitude_step, Latitude_step* respectively represent the magnification level, the difference value of longitude each tile covered, the difference value of latitude each tile covered).

$$ZoomLevel = \{1,2,3,4,5,6,7,8,9...n\} \quad (1)$$

$$Longitude\_step = \{90,40,20,10,5,2,1,0.5,0.2...n\} \quad (2)$$

$$Latitude\_step = \{90,40,20,10,5,2,1,0.5,0.2...n\} \quad (3)$$

B. Simple Index

Each tile has a unique file name, and all the tiles are stored in hierarchical file system directories using the zoom level as the directory name. When a client sends a request to get the tile from caches server, the server can directly retrieve the exact tile without the requirement of querying any database, which would decrease the response time of the server. We assumed each tile named *x, y* and *request_longitude, request_latitude* represent the value of coordinate which client requested, and then we can get the formula as below (the *floor* function returns the largest integer value that is less than or equal to the special parameter）:

$$x = floor(\frac{request\_longitude}{longitude\_step[n]}) \quad (4)$$

$$y = floor(\frac{request\_latitude}{latitude\_step[n]}) \quad (5)$$

C. Condition Constraint

The number of tile file will grow exponentially along with the increase of map magnification level, but most operating system has limitations for numbers of files in a single directory. One more efficient mechanism for directory organization and tiles indexing needs to be applied. In our study, we adopt an index algorithm based on grids to reorganize the tile caches. In every zoom level, the whole map would be split into many grids by multiple steps horizontally and vertically. Each grid is a square matrix which covers the same number of tiles in horizontal and vertical orientation.

D. Indexing Algorithm

In the reorganizing process, number 10 can be taken as the steps' multiple. We call the multiple values the step of grid (defined with *Gridstep* in Formula (6)), and then can get the formula as below:

$$Gridstep = \{10,10,10,10,10,10,10,10,10...n\} \quad (6)$$

We use quotient of the division between x (*or* y mentioned above) and $Gridstep[n]$ as index number which used as the further hierarchical directory name. We assume *folder_x*, *foler_y* respectively represent the directory name and *index_x*, *index_y represent* the index number of each map tile. Then each tile can be indexed from the formula as below:

$$folder\_x = floor(\frac{x}{Gridstep[n]}) \quad (7)$$

$$folder\_y = floor(\frac{y}{Gridstep[n]}) \quad (8)$$

$$index\_x = x \bmod Gridstep[n] \quad (9)$$

$$index\_y = y \bmod Gridstep[n] \quad (10)$$

With the increase of the zoom level, the number of tiles will grow exponentially and the efficient of indexing would be impacted if *Gridsetp* keep unchanged. In the implementation of application, the value of $Gridstep[n]$ can be changed to be appropriate according to the different zoom level. Take the world map as an example, *Gridstep* can be defined as below:

$$Gridstep = \{10,10,10,10,10,10,10,10,10,10,50...\} \quad (11)$$

For example, if this point's location is 114°15'4" E, 30°43' 50"N. According to the formula series mentioned above, we can figure out the indexing number of the map tile at level 14 as below:

$$index\_x = floor(\frac{request\_longitude}{longitude\_step[n]}) \bmod (Gridstep[n])$$

$$= floor(\frac{114.251}{longitude\_step[14]}) \bmod (Gridstep[14])$$

$$=1 \quad (12)$$

$$index\_y = floor(\frac{request\_latitude}{latitude\_step[n]}) \bmod (Gridstep[n])$$

$$= floor(\frac{30.725}{longitude\_step[14]}) \bmod (Gridstep[14])$$

=45 (13)

We can further figure out the special location on this tile (1, 45.png) according to the ratio of width-to-length and the step-length of longitude and latitude this tile covers. Then the program would adjust the tiles' location in the view window and symbolize this special point to show on the center of the visualized map.

E. *Pre-generated Processing*

Before all the tile caches pre-generated, the spatial data should be transferred into predetermined coordinate system. For realizing the practicability and aesthetics of the map, the map also should be rendered before split. The rendering process can be done in any GIS software, such as ArcMap, MapInfo etc. then a split program should be executed to accomplish the pre-generated processing on consistent platform. Take ArcIMS Server as an example to introduce the process of pre-generating. Fig. 2 and Fig. 3 show a typical request and response for ESRI map service. In the service of ArcIMS Server, the communication between the client and the server is based on ArcXML. These XML messages are transported over HTTP and the tile caches are not directly returned in the response. ArcIMS Server sends back a URL to the client instead, which allows the server to communicate in a non-blocking fashion with the client. This is convenient as map images are typically created on demand, and it can take several seconds to generate the tile actually [7].

```
<? xml version= "1.0" encoding= "UTF-8" ?>
<ARCXML version= "1.1" >
  <REQUEST>
    <GET_IMAGE>
      <PROPERTIES>
        <ENVELOPE minx= "114.50" miny= "30.48" maxx= "114.55" maxy= "
30.52" />
        <IMAGESIZE width= "300" height= "240" >
      </PROPERTIES>
    </GET_IMAGE>
  </REQUEST>
</ARCXML>
```

Figure 2. A sample request to ArcIMS Server

```
<? xml version= "1.0" encoding= "UTF-8" ?>
<ARCXML version= "1.1" >
    <RESPONSE>
    <IMAGE>
    <ENVELOPE minx= "114.50" miny= "30.48" maxx= "114.55" maxy= "30.52"
/>
    <OUTPUT
    url= "http://caches.vrwuhan.com/caches/wuhan_CACHES5864475618.png/>
    </IMAGE>
    </RESPONSE>
</ARCXML>
```

Figure 3. The response of the request in Figure 2.

F. *Storing and Indexing Tile Caches*

When the tile caches are generated, the file of each tile must be renamed and moved to predetermine directory on map server according to the index algorithm mentioned above. Because all of the tiles are stored in hierarchical directory, each tile needs to be copied to map server from ArcIMS Server caches directory immediately when it is generated and all tile caches should be reorganized on map server. All tile caches form a pyramidic data structure [8] with the whole globe covered by one tile, in the next zoom level with 4 tiles, then 25 tiles, etc. In our study, the tile caches for zoom level 14 are as shown in Fig. 4. We assumed that a point has the coordinate (*longitude*, *latitude*), the index way to locate this point can be drawn as follow formula (*n, folder_x, folder_y, index_x, index_y* respectively represent zoom level, index number of folder and index number of tile):

Tile_path = http://maps.vrwuhan.com /n/folder_x,folder_y/index_x,index_y.png

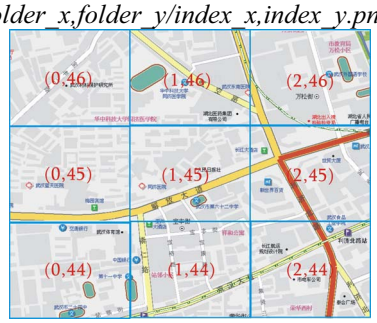
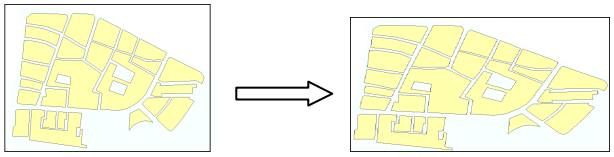


Figure 4. Tile caches of zoom level 14 (www.vrwuhan.com)

IV. Deformation Problems

On the process of pre-generating tile caches, all the tiles are in the similar size (e.g. 256×256 pix) and each tile covers same longitude difference and latitude difference. But earth is an irregular spheroid which determines the distance of one segment with the same longitude difference or latitude difference is different. Through the Geometric computing, it is appropriate to adopt this approach when the regions near the equator. In the Mid-latitudes area, the deformation problem is obvious. And the deformation would be much larger when the latitude value getting higher. Fig. 5 shows the houses deformation using this approach in Wuhan City, China.



Original shape deformed shape

Figure 5. Houses deformation in Wuhan, China

The reason which causes the deformation is the different distances which *Longitude_step* and *Latitude_step* represent. Assumed *radius_earth, dis_perlongitude, dis_perlatitude, α* respectively represent the radius of earth, the distance one longitude covers, the distance one latitude covers and latitude. From the Fig. 6, the formula can be drawn as below:

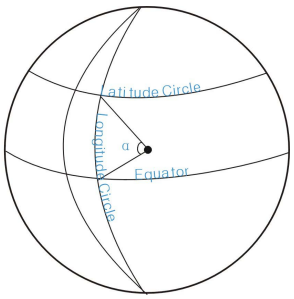


Figure 6. Longitude and latitude of the earth

$$dis\_perlatitude = \frac{\pi \times radius\_earth}{180} \quad (14)$$

$$dis\_perlongitude = \frac{\pi \times radius\_earth}{180} \times \cos(\alpha) \quad (15)$$

From the formula above we can get a conclusion that the ratios of distance_perlongitude and distance_perlatitude have nothing to do with the radius of earth. And it has only the relation to the value of the latitude. So, there are two ways to correct the deformation: one is to modify the ratio of the width-to-length to each pre-generated tile, the other is to modify the ratio of the *distance_perlongitude* and *distance_perlatitude* each tile covers. We take Wuhan City (114° 22'54" E, 30° 33'40"N) as an example to illustrate the correcting way in detail.

In the first way, because $\alpha = 30° 33'40"$, $\cos(\alpha)=0.861$, the ratio of the width-to-length of each tile should be 0.861. For the convenience of computing, 0.8 can be adopted to resolve the deformation problem. For example, each tile can be resized 300×240 pix when it is generated.

The second way also adopts 0.8 as the ratio of *Latitude_step* and *Longitude_step* and it can be implemented by modifying the *Latitude_step* which is redefined as below:

$$Latitude\_step = \{90\times0.8,40\times0.8,20\times0.8,10\times0.8, 5\times0.8,2\times0.8,1\times0.8,0.5\times0.8...\} \quad (16)$$

The amended parameters are different in different area. So, it is necessary to zone according to the latitude, the more number was zoned, the less deformation would be resulted.

V. Implementation in Wuhan City

In our study, we take Wuhan City as an example to develop this on-line mapping web application. Wuhan City covers 113°41'~ 115°05'E，29°58'~ 31°22'N. We divide the whole map into 11 zoom levels. The amended parameter adopts 0.8. The longitude and latitude of each tile, the numbers of tile caches, scales are as below (Table 1) and the snapshot of implementation is shown in Fig. 7.

TABLE I. Grading of Wuhan City using tile caches technique

Zoom level	Longitude-step	Latitude-step	Number	Scale
1	1	0. 8	12	1, 571, 167
2	0. 5	0. 4	20	785, 584
3	0. 2	0. 16	80	314, 233
4	0. 1	0. 08	300	157, 717
5	0. 05	0. 04	1, 110	78, 558
6	0. 02	0. 0016	6, 482	31, 423
7	0. 01	0. 008	25, 242	15, 711
8	0. 005	0. 004	100, 688	7, 856
9	0. 002	0. 0016	623, 063	3, 142
10	0. 001	0. 0008	2, 492, 252	1, 571
11	0. 0005	0. 0004	9, 969, 008	785

Figure 7. Snapshot of implementation in www.vrwuhan.com

VI. Conclusion

This paper proposes an efficient mechanism for organizing and indexing tile caches on map server based on grid index and further analyses the deformation of shape using this technique. This paper also gives two ways to solve the problem through modifying the ratio of width-to-length to each tile or of longitude-step and latitude-step. Finally, this paper implements the on-line mapping web application using the tile-cache technique in Wuhan City, China. The following work is to build a mathematical model to fit the different amended parameter in different regions with WGS-84.

VII. Acknowledgment

This work was supported by Wuhan Construction Commission and Wuhan construction research program 200724. The authors would also like to thank Mr. Peng Qingshan and Mr. Jing Zhiqiang for their contributions.

References

[1] Google Maps API Documentation; Map Overlays: Custom Map Types; http://code.google.com/apis/maps/documentation/overlays.html#Custom_Map_Types

[2] Microsoft Virtual Earth Tile System; Virtual Earth Developer Center; http://msdn.microsoft.com/en-us/library/bb259689.aspx

[3] Li Haiting, Fei Lifan, Peng Qingshan, Li Yanhong, "The Application of Pre-generation Thinking in Geographical Information Services". Journal of Geomatics, 2009, 34(1):31-32.

[4] Sterling Quinn, Jeremy Bartley, David Wilcox. “Implementing and Optimizing ArcGIS Server Map Caches”, 2008 ESRI Developer Summit, March 17-20,2008.Palm Springs, CA

[5] Michael Potmesil, Bell Laboratories, Lucent Technologies. “Maps Alive: Viewing Geospatial Information on the WWW”. Sixth International World Wide Web Conference. April 7-11, 1997 Santa Clara, California USA

[6] http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14254/geor_intro.htm

[7] Zao Liu, Marlon E. Pierce, Geoffrey C. Fox, Neil Devadasan. “Community Grids Lab Implementing a Caching and Tiling Map Server: a Web 2.0 Case Study”. Collaborative Technologies and Systems, 2007. CTS 2007. pp. 247-256

[8] Pu Huan, Jing Tao, Li Xinghua, Zhang Sidong. “Pyramid Data Structure of GIS and Display Algorithm for PDA”. Journal of Beijing Jiaotong University, 2005.04, 29(2):72-75