COMPARISON OF TILE SERVER DESIGN APPROACHES FOR 3-D GEO-VISUALIZATION

Tao Wang[a], Jianhua Gong[b]

[a]State Key Laboratory of Remote Sensing Science ,Institute of remote sensing applications chinese academy of sciences - wangtao_temp@yahoo.com.cn
[b]State Key Laboratory of Remote Sensing Science ,Institute of remote sensing applications chinese academy of sciences - jhgong@irsa.ac.cn

Commission VI, WG VI/4

KEY WORDS: Tile server, Wms, Visualization

ABSTRACT:

In order to display large-scale maps on the Internet, it is necessary to divide the huge spatial data into small tiles. And servers are needed to support the display across network. There are two kinds of tile server. One is tile server, which organizes the pre-rendered tiles on the server. Another one is MapServer, which generates the display tile. In this article, different servers are compared in efficiency. The open source software world wind is used to analyze the capacity of the two kinds of servers, and experiment results show that tile server spent more time than Mapserver, and it is suggested the Mapserver may have been optimized.

1. INTRODUCTION

When displaying large-scale maps on personal and mobile computers, it is necessary to divide the huge spatial data into small tiles. There are many important characteristics of the tileset file format.

A tileset map can be used in memory-mapped mode on computers having either “little-endian” or “big-endian” architectures.

The tileset file format accommodates either lossy or lossless compression of bitmaps, or a combination of both.

Tilesets can be used effectively on systems with very little memory. Since the component tile spatial clustering is based on the morton square schema, fastest possible navigation within large external files is assured.

Once a tileset has been assembled, all geopositioning and navigational functions use only integer arithmetic. Despite this, geometry computations are both global in scope and of centimetre-range precision. The tileset file format therefore is well suited for use in devices having no floating-point hardware.

In order to display these tiles, tile server is needed. There is a kind of tile server called on demand tile server, on demand tile server is capable of generating tiles. ArcIMS doesn’t serve tiles, it renders complete maps and serves them. The Virtual Earth and Google maps Internet services are tile services because they serve up pre-rendered tiles. ArcIMS renders maps on the fly, while these new consumer mapping service don’t.

MapDotNet Server let users combine dynamically rendered tile overlays based on local spatial data sets with pre-rendered Virtual base maps and Web 2.0 functionality. MapDotNet Server can optionally cache tiles so that unchanged data doesn’t have to be queried and rendered repeatedly.

After researching Google Earth Enterprise, which includes Google Earth Fusion, and MapCruncher for Virtual Earth, we’re finding out that the ability to serve tiles on demand that are rendered at the time of the request, and overlay these tiles on a Virtual Earth or Google Maps base map is unique. Google Earth Enterprise LT offers some of this functionality, but only for the fat Google Earth 3D client. It’s actually uncertain that Google Earth Enterprise LT can serve tiles that are generated on demand, but it can overlay pre-rendered tiles that are based on your own data in the Google Earth fat client.

There’s an almost unfathomable number of options out there for people that want to visualize spatial data or perform geographic analysis and functions.

OpenLayers is an interesting Internet based tile service much like Virtual Earth and Google Maps. OpenLayers allows you to add layers from a WMS service. A WMS service will render tiles on demand. you could theoretically use WMS Tile Caching with OpenLayers and have a high performance solution like MapDotNet Server.

There’s not much theory there, EduGIS.nl used mapbuilder and UMN MapServer to serve tiles in a WMS-C style (it was written before the WMS-C spec was finalized). It uses apache to cache the tiles and is as quick as google maps. The only problem of this solution is the manageability of the cached data. It is not very easy to remove those tiles of the cache that need updating. The current approach is very much a render-on-demand: if it is in the cache serve for the cache, if not, render the tile from the data and put it in the cache.

Spatial data visualization and Enterprise GIS are being transformed rapidly by these technologies, and they are all making it into the mainstream. An on demand tile server that can leverage Virtual Earth base maps and cache local tiles for better performance provides organizations with the Web2.0 user experience they are looking for alongside the depth of real GIS capabilities and resources that they’ve been working on building for the past 2 decades.

2. PRINCILE OF TIEL SERVER AND MAPSERVER

In the open source software World Wind, there are two kinds of tile server. One is based on the MapServer, another is tile server. Here we'll discuss their difference and how to build them.

Although World Wind provides good map and aerial coverage of most areas, you are still the best authority on data availability and data quality for your own needs. Whether you have advanced imagery, engineering plans, or transportation maps, serving this information is often a challenging problem. You can take advantage of World Wind's Tile Layers to create your own imagery server that integrates fully with the navigation methods, data and tools built into World Wind.

In this article, we will look at the techniques you need to serve up your own image data using World Wind technology.

The most difficult challenge in building your own tile server lies in matching up your data with the World Wind map data. This process is often challenging, because you have to overcome the joined problems of projection and registration. That is, you need to stretch your image to fit World Wind's Plate Carree projection, and you also have to determine latitude and longitude boundaries of your image.

Behind the scenes, World Wind Tile Server divides up each map and map style into a set of 512*512 pixel tiles. The tiles are stitched together according to a naming schema that is based on the images position and relative degree of resolution. Each tile is numbered according to where it is with respect to a 4 square quadrant system.

Tile Server use what is defined as "Level Zero Tile Size" to determine how large (in decimal degrees) each tile is in width and height (all tiles are square). A standardized level zero tile size is under consideration but is not yet implemented, but it must divide into 180 evenly. The level zero tile size is simply the distance traveled in degrees from one side of a tile to the next side. In the NLT Landsat 7 datasets, the LZTS is normally set as 2.25 degrees. For each increase in level, the LZTS calculated using this formula: LZTS*0.5^level. What this does effectively decreases the tile size by one half for each increase in level. Where there was one tile, there are now four.

To find the coordinates of the bottom left corner of the next higher tile you would add the appropriate tile size for the level you are in, to the latitude and/or longitude. Adding to latitude would of course increase the Y value, longitude increasing the X value. Figure 1 demonstrates bringing both the concept of X/Y (now displayed as (x,y)) and latitude/longitude. Figure 2 has a LZTS of 2.25 degrees.

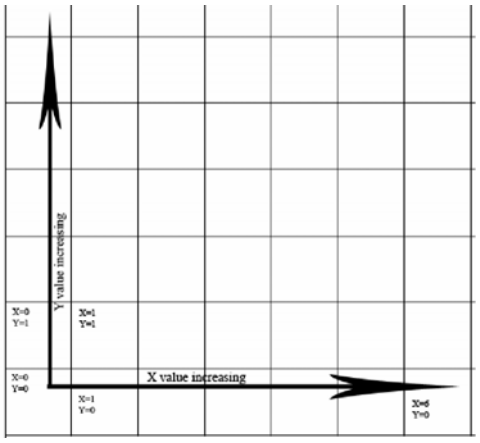

Figure 1 World wind tile structure

To prepare your own tiles, you will need to cut your image into 512*512 pixel chunks and create a meaningful naming schema. You will also have to decide how you want to host your images. If you intend to follow World Wind Tile Server naming schema, you will need to figure "register" your images with respect to the World Wind tiles. That is, your tiles must share bounding boxes with WW tiles.

World Wind Tile Server uses a multi-resolution layering technique that shows progressively more detail as a user zooms in to various locations. To do this, World Wind stores multiple copies of the same map at the successively higher resolutions. Using a map in WGS 84 projection of the earth as an example, at layer 0, world wind breaks down the image into 50 tiles, each at 36 * 36 segments, see following figure. Layer 1 increases this resolution by a factor of 4, meaning that the same image of Earth is now broken down into 18 * 18 segments, resulting in 200 tiles of information. At layer 2, the resolution becomes 9 * 9 for 800 tiles, Layer 3 is 4.5 * 4.5 and 3200 tiles, etc.

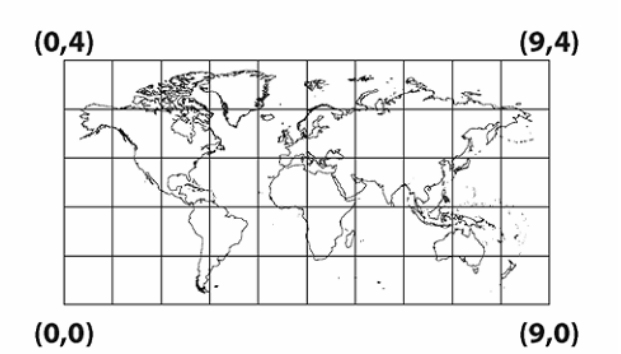

Figure 2 Coordinate system of world wind

As a user zooms in to various locations, World Wind requests the necessary data from servers across the internet, downloading only the information that is required to display the view requested by the user.

In its simplest form, World Wind directly requests tiled imagery (images of size 512*512 in jpeg and other formats) from a server using a simple NASA custom HTTP request protocol. The client issues a simple request with the following parameters:

T=dataset &X= column &Y= row &L=resolution level.
Here, the column, row and level parameters refer to the location and size of the geographically referenced region on the worldmap in a simple geographic projection.

Proper elevation data and image coverage are requested for the specific resolution layer and view point. World Wind caches the requested data locally and therefore can seamlessly transition from one layer to the next, revealing successively more detail as the user navigate around the World:

Equally well supported is the Web Mapping Service 1.3 protocol specification from the Open Geospatial Consortium. WMS is a broadly adopted standard. World Wind easily operates with servers based on this standard.

In order to have a good understand of WMS, we use the open source software MapServer to build the tile server.

MapServer is an open source development environment for building spatially-enabled web mapping applications and services. It is fast, flexible, reliable and can be integratated into just about any GIS environment. Originally developed at the University of Minnesota, MapServer is now maintained by developers around the world.

MapServer run on all major operating systems and will work with almost any web server. MapServer features MapScript, a powerful scripting environment that supports many popular languages including PHP, Python, Perl, C# and Java. Using MapScript makes it fast and easy to build complex geospatial web applications[4].

MapServer works behind a web server application. The web server receives requests for maps and passes them to MapServer to create. MapServer generates the requested map image and hands it to the web server, which transmits it back to the user. The above figure shows how the users interacts with the web server which, in turn, makes requests to the MapServer program. Fig 3 shows wms server architecture[4].

MapServer's primary function is reading data from various sources and pulling these layers together into a graphic file, also known as the map image. One layer may be a satellite image, another outline of your country or points showing a major city. Each layer is overlaid or drawn on top of the others and then printed into a web-friendly graphic for the user to see. A good example of the results of the overlapping and mapping process can be seen. You can see a satellite image(from a remote server), road lines, and city locations; the city labels are dynamically generated by MapServer. Fig 3 shows the basic operation of a Mapserver's application[4].

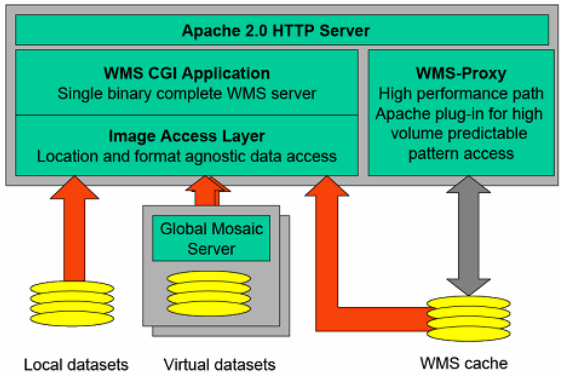

Figure 3 WMS server architecture

This drawing process occurs each time a request for a new map is made to MapServer, for instance, when a user zooms into the map for a closer look. This process also occurs when a user manually requests a redraw, such as when the content of one data layer changes, and the user wants to see the change.

MapServer creates map images from spatial information stored in digital format. It can handle both vector and raster data. MapServer can render over 20 different vector data formats, including shapefiles, PostGIS and ArcSDE geometries, OPeNDAP, Arc/Info coverages, and Census TIGER files[4].

MapServer can operate in two different modes: CGI and MapScript. In CGI mode, MapServer functions in a web server environment as a CGI script. This is easy to set up and produces a fast, straightforward application. In MapScript mode, the MapServer API is accessible from perl, Python, or PHP. The MapScript interface allows for a flexible, feature-rich application that can still take advantage of MapServer's templating facilities[4].

MapServer is template based. When first executed in response to a web request, it reads a configuration file (called the mapfile) that describes the layers and other components of the map. It then draws and saves the map. Next, it reads one or more HTML template files that are identified in the mapfile[4].

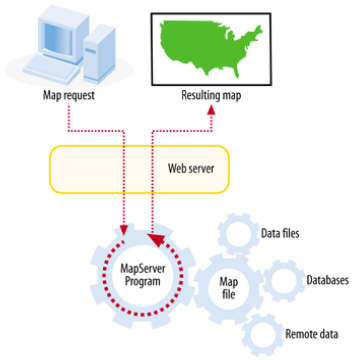

Figure 4 Diagram showing the basic operation of a MapServer application.

Following is a list of the required GetMap parameters according to the WMS spec:

VERSION=version: Request version

REQUEST=GetMap: Request name

LAYERS=layer_list: Comma-separated list of one or more map layers. Optional if SLD parameter is present.

STYLES=style_list: Comma-separated list of one rendering style per requested layer. Optional if SLD parameter is present. NOTE that MapServer does not support named styles, so most times you would specify "STYLE=" with an empty value. MapServer does support STYLE when used with an SLD.

SRS=namespace:identifier: Spatial Reference System.

BBOX=minx,miny,maxx,maxy: Bounding box corners (lower left, upper right) in SRS units.

WIDTH=output_width: Width in pixels of map picture.

HEIGHT=output_height: Height in pixels of map picture.

FORMAT=output_format: Output format of map.

A valid example from WorldWind would therefore be:

Url:http://wms.jpl.nasa.gov/wms.cgi?request=GetMap&layers=global_mosaic&srs=EPSG:4326&width=512&height=512&bbox=114,28,116,36,30&format=image/jpeg&version=1.1.1&styles=visual

3. EXPERIMENT

NASA uses just a filesystem backend for almost all of WW's datasets, including elevation. TerraServer, which pulls the tiles from a SQL database. OnEarth with a WMS + a smart CGI script + a cache for WW. The Free Earth Foundation server for the Zoomit Dataset and others using a packed tile schema and some simple PHP.

We can't analyze the difference of map, so we use the world wind to analyze.

1.From World Wind, we compare the time of tile server and MapServer. Figure5 shows downloading time from the two different server, Following data are choosed for testing.

2.Cubed ESAT World Landsat7 Mosaic NASA derived global 15 meters per pixel satellite image mosaic, donated and processed by I-Cubed using equipment from Isilon Systems. (blue line)

3.OnEarth 15m Global Mosaic, pseudocolor. (red line)

4.OnEarth 15m Global Mosaic, visual. (yellow line)
Transferation time of different levels are calculated. Only five levels are included.

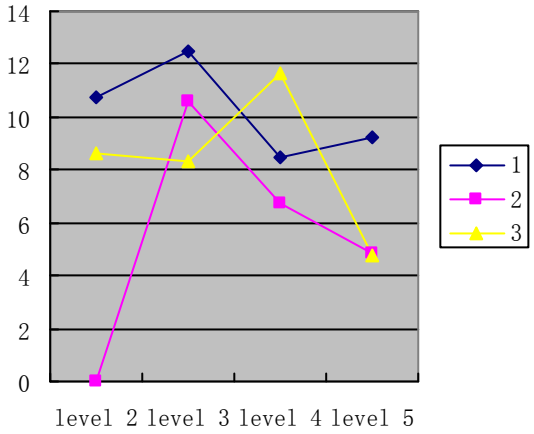

Figure 5 Image downloading time from different servers

4. CONCLUSION

Instead of using an overlay complex format (for serving predefined chunks that is) WorldWind opted to go with pre-rendered, pre-defined tiles that are just stored in the file system, and can be served to the client with no further processing server side. This reduces the strain on the server incredibly. Mapserver can give different tile size according to the request. But from the statistics, tile server spent more time than Mapserver. So Mapserver may have been optimized. A deep research should be done to have a good understanding of Mapserver.

ACKNOWLEDGEMENTS

This research is partially supported by National High-tech R&D Program (863 Program) (2006AA12Z204), the Key Knowledge Innovative Sub-Project of Chinese Academy of Sciences (Kzcx2-yw-126-01), and National Basic Research Program of China, 973 Program, No. 2007CB714402.

REFERENCE

[1] Bill Kropla, 2005, Beginning Mapserver: Open Source GIS Development. Apress.

[2]Tyler Mitchell, 2005, Web Mapping illustrated. O'Reilly.

[3]Schuyler Erle, 2005, Mapping Hacks. O'Reilly.

[4]OGC,http://www.opengeospatial.org/(accessed January 2007)

[5]David G.Bell, Frank Kuehnel, NASA World Wind: Opensource GIS for Mission Operations.

[8]Open source community portal, http://www.worldwindcentral.com. (accessed Apr 2008)

[9]OpenGIS, Beaujardiere, J. (editor)(2004). OGC Web Map Service Interface. OGC 03-109rl.