A Study of Real Time Raster Graphic Visualization Based on Big Data Technology

Zhi-heng GAO
Research Institute of China Telecom Co., Ltd., Guangzhou 510630, China

Keywords: Big data, HBase, Raster graphics, Graphics visualization.

Abstract. The grid sizes were designed to provide best quality for raster graphics. The grid data was regionally grouped into blocks and stored by partition, providing a data structure suitable for distributed parallel processing. The whole procedure from data query to grid block sub-graphic generation was fully parallel processed by an independently developed cluster, achieving high performance visualization of pixel-level raster graphics under full-HD screen resolution. The solution provides reference for the construction of similar systems.

Introduction
In GIS, there are two primary types of spatial data: vector and raster data. Vector graphics comprise vertices and paths. Raster graphics are made up of pixels (also referred to as grid cells but not necessarily identical). They are usually regularly-spaced and square. Raster graphics often look pixelated because each pixel has its own value or class[1]. The advantage that vector graphics have over raster ones is that they can be resized without any visible negative impacts. This is mainly due to the mathematical formulas, which allow the image to hold its form, as the formula dictates how the image is rendered. Raster graphics are mainly used for high density images, or images with many different colors, as each pixel can have a different color, which can be overlapped for different colors and shapes.

There are two methods to visualize a raster graphic. One is pre-slicing, which is to pre-generate small sliced graph blocks according to various map scales in the studied geographical area, and directly load relevant blocks when users visit some area. Because the graphics have been pre-generated, the response of this method is very fast. However, when there are many kinds of indicators, each indicator needs a separate set of graphs, so a very large storage space is required; moreover, it cannot meet the user-oriented customized requirements such as individual index segmentation and personalized coloring.

The other method is dynamic rendering, which calculates the location of the pixels according to the grids to be displayed, and calculates the corresponding color of each grid/pixel according to the grid index level, and then visualizes it. This method can flexibly adjust the visual effects of color system and index segmentation. However, when the amount of grid data is tremendous, the system performance overhead is huge. In order to provide a faster visualization response, the grid is often designed roughly to reduce the data needed to be retrieved, so the graphic precision is not high. This paper studies how to build an efficient and high-precision raster graphics real-time visualization system based on big data technology.

Design of Grid Cell Sizes
Data of raster graphics is stored by grid cells. When the map plane is uniformly divided by latitude and longitude lines, grid cells are formed. There are different zoom levels for map, accordingly there are different sizes for grid cells.

In many scenarios, GPS is used for positioning. GPS commits to broadcasting the signal in space with a global average user range error (URE) of \( \leq 7.8 \text{m} \), with 95% probability. However, URE is not user accuracy, which depends on a combination of satellite geometry, URE, and local factors such as

215
signal blockage, atmospheric conditions, and receiver design features/quality[2]. So, we use 20m as minimum grid size. Grid sizes are designed to double step by step from 20 meters to form such a series as [20,40,80,160,320...], so that each level can be directly aggregated from the previous level, which means reducing the amount of computation.

For a given zoom level, different grid sizes could be used to generate the raster graphic, the smaller the grid size, the better the graphic quality. However, as all relevant grids are finally mapped to pixels, when the grid size is less than the pixel distance, there would be no help for graphic quality no matter how small the grid size is. So the maximum of the grid sizes which are smaller than the pixel distance is chosen for that zoom level. As a result, the corresponding relation between map zoom level and grid size is shown as Tab. 1.

<table>
<thead>
<tr>
<th>Zoom Level</th>
<th>Scale</th>
<th>Pixel Distance (m)</th>
<th>Grid Size (m)</th>
<th>Estimated Grid Count in Full-HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0</td>
<td>20</td>
<td>0.42</td>
<td>914</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>20</td>
<td>0.82</td>
<td>3486</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>20</td>
<td>1.64</td>
<td>13943</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>20</td>
<td>3.28</td>
<td>55772</td>
</tr>
<tr>
<td>200</td>
<td>4</td>
<td>20</td>
<td>6.41</td>
<td>213001</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>20</td>
<td>12.82</td>
<td>852003</td>
</tr>
<tr>
<td>500</td>
<td>6</td>
<td>20</td>
<td>25.64</td>
<td>3408011</td>
</tr>
<tr>
<td>1000</td>
<td>7</td>
<td>20</td>
<td>50.51</td>
<td>3306433</td>
</tr>
<tr>
<td>2000</td>
<td>8</td>
<td>20</td>
<td>101.01</td>
<td>3305779</td>
</tr>
<tr>
<td>5000</td>
<td>9</td>
<td>20</td>
<td>202.02</td>
<td>3305779</td>
</tr>
<tr>
<td>10000</td>
<td>10</td>
<td>20</td>
<td>423.73</td>
<td>3635829</td>
</tr>
<tr>
<td>20000</td>
<td>11</td>
<td>20</td>
<td>847.46</td>
<td>3635829</td>
</tr>
<tr>
<td>50000</td>
<td>12</td>
<td>1280</td>
<td>1694.92</td>
<td>3635829</td>
</tr>
</tbody>
</table>

Grouping of Grid Cells

Tab. 1 shows that huge amount of grid cells will be involved to generate a full screen graphic in full HD resolution. When grid size is as small as 20m, there are tremendous grids. For example, a south China province like Guangdong holds more than 450 million such grids and data accumulates day by day, that is, more than 13.5 billion grid records a month. We use HBase for grid data query. HBase is an open source database with the characteristics of supporting massive data, column storage, distributed scalability, high reliability and high performance. It can provide millisecond-level fast response for data retrieval[3].

In an HBase table, data, which is divided into row keys and values, is sorted and stored by row keys. Fig. 1 shows the response time curves of enumeration and batch retrieval, respectively, according to different grid quantities. Enumeration retrieval means listing all keys of grids for querying, while batch retrieval only gives the starting and ending range of keys. As shown in Fig. 1, when the grid data is less than 30,000, the response times of the two retrieval methods are of little difference, but when grid quantity increases to 100,000 and even 300,000, time of enumeration retrieval increases sharply, meanwhile batch retrieval remains much more efficient.

If grid data were simply stored by grid number as row key, it has to be queried via enumeration. In order to provide better performance, we group a matrix of k*j(where k,j are natural numbers) grids into a block. Every block has a unique key and data of all grids in the same block are concatenated into
the value. By this means, grids are batch retrieved via a whole block. That is, the search workload can be reduced to \(1/(k\times j)\) of the original.

![Figure 1. Response Time for enumeration and batch retrieval.](image)

**Partitioning and Parallel Processing**

In the case of full-screen display, when the grid size is less than the pixel distance, the number of grid blocks is as many as hundreds. To further improve the processing efficiency, a distributed computing architecture is introduced, and multiple computing nodes are used to parallel process the tasks of sending query request and generation sub-graphics of blocks.

The row keys of HBase tables should be carefully designed to avoid read/write operations being concentrated on a region server at the same time period, which results in load imbalance[4]. When the grid block key is simply composed of date and grid block number, the grid blocks relevant to some target area are basically concentrated in one or two region servers, which can easily cause local overheating in HBase cluster. For this reason, pre-partitioning should be used to force adjacent grid blocks to be distributed to different region servers, so as to avoid computing workload imbalance in HBase cluster [5]. In our case, the pre-partitioning code is designed as \(P=(N\times j+M)\%(j\times k)\), where \(j, k\) are natural numbers, \(M\) is longitude direction order number of the grid block and \(N\) is latitude direction order number. This design ensures that \(P\) will not be repeated in any matrix less than or equal to \(j\times k\) grid blocks, thus achieving good distribution.

The technical architecture of the real time raster graphic visualization system is shown as Fig. 2. The Data Storage layer is HDFS, responsible for storing HBase tables. HBase is used for high performance grid block data retrieval. An independently developed distributed cluster for data query and sub-graphics generation is placed between Visualization Service layer and Data Retrieval layer. Master in the cluster is responsible for: identifying the keys of all relevant grid blocks; assigning tasks to all active Workers on average according to keys; receiving all sub-graphics returned by Workers and then passing them to the Visualization Service layer. Workers are responsible for: submitting queries to the HBase cluster (enumerating all keys assigned by Master); generating sub-graphics of each grid block based on the topological relation between the grid block and the target area; returning all generated sub-graphics to Master. There should be multiple Master nodes deployed to provide high availability, and the entire cluster is integrated into Zookeeper for unified management. Visualization Service layer is developed in Java Web, it simply receives visualization requests and bypass all the
sub-graphics to Browser layer for rendering. All the workload is place on the data query and sub-graphic generation cluster. As a result, the whole system could achieve very high performance, even though there are millions of grids involved in each visualization request.

Based on the above solution, in our production system, a grid block consists of 64*64 grids, j and k in partition equation are both 8, and the distributed cluster contains 64 nodes, the raster graphic visualization task for a full screen or a polygon of any shape, which often involves 3 to 4 million grids, could be accomplished within 3 seconds. Fig. 3 is a partial screen shot of the visualization effect.

Summary

The grid sizes were designed to provide best quality for raster graphics. The grid data was regionally grouped into blocks and stored by partition, providing a data structure suitable for distributed parallel processing. The whole procedure from data query to grid block sub-graphic generation was fully parallel processed by an independently developed cluster, achieving high performance visualization.
of pixel-level raster graphics under full-HD screen resolution. The solution provides reference for the construction of similar systems.

Acknowledgement
This research was financially supported by China Telecom.

References