Efficient Histogramming of Large-Scale Geospatial Rasters in Supporting of Web-Based Queries

Simin You
Dept. of Computer Science
CUNY Graduate Center
New York, NY, 10016
syou@gc.cuny.edu

Jianting Zhang
Dept. of Computer Science
City College of New York
New York City, NY, 10031
jzhang@cs.ccny.cuny.edu

ABSTRACT

This demonstration paper reports our design and implementation of a Web-based system that supports efficient histogramming of large-scale geospatial rasters using either rectangular or arbitrarily shaped polygons interactively drawn by users at the client side. By combining two performance boosting techniques, namely materialized histograms and compressed tiles, experiments using 50 random Region of Interests (ROIs) show that our approach is several times faster than straightforwardly scanning all raster tiles that intersect with the ROIs.

Keywords

Histogram, Query Optimization, Geospatial Raster, Web-GIS

1. INTRODUCTION AND MOTIVATIONS

Histograms are widely used in many applications such as image processing, data mining and database applications. Most commercial and open source GIS, such as ESRI ArcGIS [1] and GDAL [2], have provided the functionality to generate histograms. Unfortunately, they do not allow dynamically specifying Region of Interests (ROIs), using either rectangles or interactively drawn arbitrary polygons. More importantly, the performance of histogramming in these systems becomes really poor on large scale geospatial rasters. While more and more GIS functionality are becoming available over the Web, to the best of our knowledge, we are not aware of existing Web-GIS that allow efficiently querying histograms of large-scale geospatial rasters over the Web with arbitrarily shaped ROIs. Towards this end, we have integrated a set of techniques and developed a Web-based system that allows users to interactively draw rectangles or any arbitrarily shaped polygons at the client side to query histograms of large-scale geospatial rasters in remote geospatial data repositories.

2. THE PROPOSED APPROACH: FRAMEWORK AND TECHNIQUES

The straightforward approach to geospatial raster histogramming is sequential scan. However, the gap between fast CPUs and slow disk I/Os is getting larger rather than smaller as modern computing technologies advance. The technological trend makes sequential scan undesirable due to the expensive I/O overheads. We have observed that, if a large raster is divided into tiled chunks, for an arbitrarily shaped ROI, only the tiles that intersect with the boundary of the ROI need to be scanned to generate histograms for the tiles dynamically while the tiles inside the ROI can use materialized histograms before the tile-level histograms are combined to derive the final histogram. Another observation is that raster tiles with high compression ratios can be compressed on disks and dynamically uncompressed in main memory to speed up disk I/O. Based on these two observations, our framework towards efficient histogramming of large-scale geospatial rasters include the following steps (1) Chunking large rasters into tiles using a suitable tile size (2) Decompose ROIs and compute tiles that are within the ROIs and tiles that intersect the boundaries of the ROIs. (3) Dynamically generate histograms for tiles that intersect the boundary of an ROI from uncompressed or compressed tiles (4) Aggregate materialized histograms of the tiles that are completely within an ROI. (5) Sum up the dynamically generated histograms and the materialized histograms. We next provide technical details on ROI decomposition which is a key component to our system.

We aim at supporting both rectangular and polygonal ROIs as shown in Fig. 1. While finding the tiles that intersect with the boundary and tiles that are within an ROI is a standard spatial query problem and can be answered by a spatial database if the tiles are explicitly represented as geometrical objects, we have chosen to provide an implementation independent of any spatial databases for the sake of efficiency and convenience.

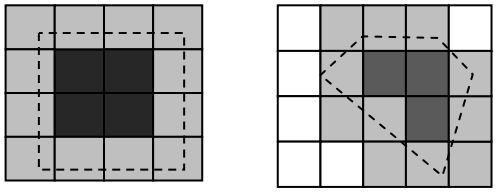

Figure 1 Decompositions of Rectangular and Polygonal ROIs

Unlike finding tiles that intersect with a rectangular ROI which is relatively straightforward to implement, finding internal and boundary tiles for arbitrarily shaped polygonal ROIs (as shown in the right of Fig 1) requires more work. We have adopted a scan line fill algorithm [3] based on a modified GDAL codebase [2] for this purpose. First we compute the bounding box of the polygon as (x1, y1, x2, y2). Based on the current zoom level at the client side, an ROI rasterization resolution S is determined. The scan lines along the y direction are thus floor(y1/S) to floor(y2/S)+1. We then loop through the scan lines and compute the intersection points of the ROI polygon with all the scan lines. The intersection points along each of the scan lines are then sorted to derive the starting and ending positions for the raster cells that are within the ROI polygon. Note that a rasterized cell of the ROI polygon correspond to S*S cells of the original geospatial raster. Assuming a tile has a size of N*N cells, we further associate a (N/S)*(N/S) 2D bitmap with each of the tiles that intersect with the ROI polygon. While looping thorough the

ACM HPDGIS'11, November 1, 2011, Chicago, IL, USA.

start and ending positions of the intersected points along each scan line, we set the corresponding bitmap cells to 1 or 0 based on whether the rasterized ROI polygon cell is within/outside of the polygon. We also keep a count of the total number of bitmap cells that have been set to 1 in the bitmap. For each bitmap, if the total number of bitmap cells that have been set to 1 is the number of the cells of the bitmap, then the associated tile is completely within the ROI. The bitmap then can be released after setting a flag at the tile level. Otherwise, the bitmap will be used to aggregate the original raster cells and derive a histogram for the tile.

While it is straightforward to rasterize an ROI polygon using the spatial tessellation of the original geospatial raster and use it for subsequent histogramming steps in a way similar to the role of mask layers used in many geospatial operations (e.g., those in ArcGIS and GRASS), we consider our implementation is more efficient due to the scaling factor of S. For ROI polygons interactively drawn by users at the client side, the accuracy of the coordinates can not go beyond a screen pixel. As such, the scaling factor will not affect the accuracy of histogramming but will improve performance. When an ROI is drawn at the detailed zoom levels at the client side, if an ROI polygon is rasterized to the original raster resolution, each intersected tiles will have to carry a full N*N bitmap with is not only computation intensive in setting the values of the bitmap cells but also requires considerable memory. In contrast, in our implementation, by using the scaling factor S, both the computation and memory consumption roughly remain constant without compromising accuracies. The reason is that, while high level ROIs often intersect with many tiles, the bitmap associated with each tile is small, i.e., (N/S)*(N/S). As users zoom in and the coordinates of the ROI polygons become more accurate, the numbers of tiles involved become smaller and the bitmap sizes associated with each tile after rasterization get larger. We can argue similarly when users zoom out. In both cases, the total computation workload and the memory footprint remain stable and thus our approach scale well. We also note that, while the algorithm runs along the scan lines, only a few or a small portion of tiles are active. Once the active scan line goes beyond the boundaries of the active tiles, the memory allocated to the bitmaps of the tiles that are completely within the polygon ROIs can be released which further makes our implementation memory efficient.

3. SUPPORTING WEB-BASE QUERIES

We have developed a Web-based prototype system to support dynamic histogram queries in major Web browsers. While many implementation options are available, we have chosen to integrate the following three technologies to realize the prototype system. First, we have developed a server program and implemented all the functionality discussed previously. Second, we have used the commercial ArcGIS Server to publish the original rasters as tiled map services so that users can visualize the underlying geospatial rasters while performing histogram queries to better explore the data in Web browsers. Third, we have used ArcGIS Flex API to work with the tiled map service for client side visualization and allow users draw arbitrarily shaped ROIs for histogram queries. Fig. 2 shows the architecture of the prototype system. For the sake of simplicity, only the structure of our server program is shown in the figure. Fig. 3 is a snapshot of our prototype system. It supports both rectangular and polygonal ROI histogram queries in addition to providing functionality reported in our previous works [4][5].

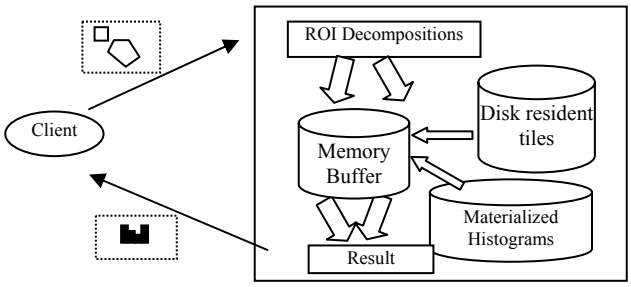

Figure 2 System Architecture

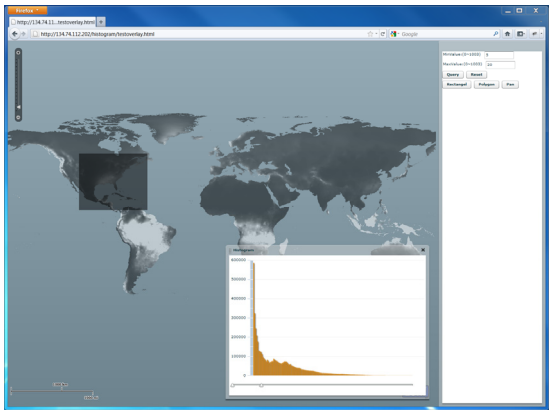

Figure 3 Snapshot of Prototype System

4. EXPERIMENTS AND RESULTS

We use the WorldClim January global 30 arc-seconds (approximately 1 kilometer) resolution precipitation data [6] in our experiments. The dataset is a 16-bit signed integer type geospatial raster and has 43200*18000 raster cells. We chunk the whole dataset with a tile size of 1024*1024 which results in 714 tiles for the dataset. The tiles are compressed as operating system files on a hard drive. Materialized histograms with 1005 bins (0-1003 and NO_DATA) are pre-generated for the tiles. Our experiments based on 50 ROIs with random centers and window sizes show that the combination of the two performance boosting techniques, i.e., materialized histograms and compressed tiles, have reduced the majority of the query response times to below 1 second. The majority of the ROI histogram queries achieved 2-6X speedup and the maximum speedup is close to 10X.

REFERENCES

1. ESRI ArcGIS. http://www.esri.com/software/arcgis/index.html.
2. GDAL: Geospatial Data Abstraction Library. http://www.gdal.org/.
3. Hearn, D. and M. P. Baker (1996). Computer Graphics, C Version (2ed). Prentice Hall.
4. Zhang J. and S. You: Dynamic Tiled Map Services: Supporting query-Based Visualization of Large-Scale Raster Geospatial data. COM.Geo Conference 2010.
5. Zhang J. and S. You: Supporting Web-Based Visual Exploration of Large-Scale Raster Geospatial Data Using Binned Min-Max Quadtree. SSDBM 2010: 379-396
6. WorldClim - Global Climate Data. http://www.worldclim.org/current.