Large Scale Analytics of Vector+Raster Big Spatial Data*

Ahmed Eldawy
Computer Science and Engineering
University of California, Riverside
eldawy@ucr.edu

Lyuye Niu
Computer Science and Engineering
University of California, Riverside
lniu001@ucr.edu

David Haynes
Program in Health Disparities
University of Minnesota, Twin Cities
dahaynes@umn.edu

Zhiba Su
Computer Science and Engineering
University of California, Riverside
zhsu@engr.ucr.edu

ABSTRACT

Significant increases in the volume of big spatial data have driven researchers and practitioners to build specialized systems to process and analyze this data. Existing efforts focus on either big raster data, e.g., remote sensing data or medical images, or big vector data, e.g., geotagged tweets or trajectories. However, when raster and vector data mix, one dataset must be converted to the other representation requiring vector-to-raster or raster-to-vector transformation before processing, which is extremely inefficient for large datasets. In this paper, we advocate a third approach that mixes the raw representations of both vector and raster data in the query processor. As a case study, we apply this to the *zonal statistics* problem, which computes the statistics over a raster layer for each polygon in a vector layer. We propose a novel method, called Scanline method, which does not require a conversion between raster and vector. Experimental evaluation on real datasets as large as 840 billion pixels shows up to three orders of magnitude speedup over the baseline methods.

CCS CONCEPTS

• **Information systems** → *Database query processing*;

KEYWORDS

Big Spatial Data, Raster, Vector, Satellite, Clipping

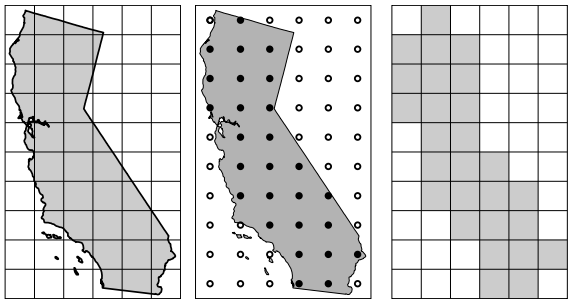
ACM Reference format:
Ahmed Eldawy, Lyuye Niu, David Haynes, and Zhiba Su. 2017. Large Scale Analytics of Vector+Raster Big Spatial Data. In *Proceedings of SIGSPATIAL'17, Los Angeles Area, CA, USA, November 7–10, 2017,* 4 pages.
https://doi.org/10.1145/3139958.3140042

1 INTRODUCTION

The rapid advancement in sensing technology resulted in a tremendous amount of spatial data collected in various domains. For example, NASA EOSDIS provided public access to more than 17 petabytes of Earth observational data, which is estimated to grow to more than 330 petabytes by 2025 [5]. Similarly, the Sentinel-1A satellite launched by the European Space Agency (ESA) collected five petabytes of data in two years and expects to receive data continuously until 2030 [6]. Additionally, the European XFEL project collects X-ray images of atoms at a rate of up to 10 petabytes per year [15].

This growth urged many researchers to build new systems for big spatial data including SpatialHadoop [3], Hadoop-GIS [1], GeoSpark [17], Simba [16], SciDB [13], RasDaMan [2], and GeoTrellis [11]. However, all these systems focus on processing either big raster data [2, 11, 13], such as satellite images, or big vector data [1, 3, 12, 14, 16, 17], such as map data or geotagged objects. While these systems are very efficient in their core function, they provide poor performance when users combine both vector and raster data in the same query.

This paper addresses the *zonal statistics* problem, which aggregates all the values from the raster layer that overlap with a set of polygons, e.g., computes the average temperature for each state in the US. This query has many applications including the study by ecologists of the effect of vegetation and temperature on human settlement [9, 10] and by geographers for analyzing terabytes of socio-economic and environmental data [7, 8]. As depicted in Figure 1, the existing methods follow one of two approaches, *vectorize* or *rasterize*. The vectorize method converts each pixel in the raster layer to a point and runs a point-in-polygon query. On the other hand, the rasterize method converts each polygon to a set of pixels in a raster layer and runs an overlay method to combine it with the



(a) Input (b) Vectorize (c) Rasterize

Figure 1: Existing approaches

*This work is supported in part by the University of California, Riverside and by the National Institutes of Health under grant NIH 5T32CA163184

SIGSPATIAL'17, November 7–10, 2017, Los Angeles Area, CA, USA

ACM ISBN 978-1-4503-5490-5/17/11…$15.00
https://doi.org/10.1145/3139958.3140042

raster layer. As the size of the raster layer increases, up to trillions of pixels, the conversion step becomes very expensive and throttles the performance of the query.

This paper proposes a new method, called *scanline*, which processes the inputs in their raw format and does not require any conversion to an intermediate form. The proposed method adopts the polygon filling algorithm from the graphics community and uses it to efficiently retrieve and process the relevant pixels in the raster layer. While previous methods suggest a preprocessing step to unify the representation of the two inputs, the proposed method shows that we can avoid that preprocessing step altogether and deal directly with the inputs in their raw format. While the proposed method can be used as a building block in parallel and distributed algorithms, this paper focuses on the single-machine implementation and we leave parallel implementation as a future work.

This research is an invitation to the SIGSPATIAL community to join our efforts in revisiting the existing operations that mix vector and raster data to develop additional optimizations that will benefit the entire community. We believe that it is important to reduce the gap between vector and raster query processing to achieve the full potential of the system's capabilities. Our proposed algorithm for the zonal statistics problem is a proof-of-concept showing how combining the two datasets in the query processor can achieve much higher performance. There is still additional effort required to extend these approaches to other problems and other platforms, such as parallel systems.

The rest of this paper is organized as follows. Sections 2 describes the baseline methods and reviews the related work. Section 3 describes the proposed scanline method. Section 4 gives details of the experimental evaluation. Finally, Section 5 concludes the paper.

2 BASELINE METHODS

This section defines the *zonal statistics* problem and gives two baseline methods based on vectorization and rasterization.

2.1 Problem Definition

The input to the problem is a raster layer r, a vector layer v, and an accumulator *acc*. The raster layer, consists of a two-dimensional matrix of values, e.g., temperature, and a transformation function, called grid-to-world ($\mathcal{G}2\mathcal{W}$). The $\mathcal{G}2\mathcal{W}$ function maps the location of each entry in the matrix, row and column, to a geographic location, e.g., latitude and longitude. The inverse of ($\mathcal{G}2\mathcal{W}$) is called ($\mathcal{W}2\mathcal{G}$). The vector layer v consists of a set of polygons which are usually disjoint and each polygon is represented as a list of points each defined by a geographic location. The accumulator *acc* is a user-provided function which takes pixel values, one at a time, and computes the statistics of interest. For example, one accumulator can compute the average value while another accumulator can compute a histogram for the values. The output is a value for the accumulator for *each polygon* in the vector layer. For example, if r represents the temperature in the world, v represents the 50 US States, and *acc* is an average accumulator, the output of this problem will be the average temperature for each state in the US.

2.2 Vectorization Method

There are several systems that focus on processing big vector data such as SpatialHadoop [3], Hadoop-GIS [1], MD-HBase [12], ESRI on Hadoop [14], GeoSpark [17], and Simba [16], among others. To run the zonal statistics operation, these systems can run a vectorization based method that consists of three steps, vectorize, spatial join, and grouped aggregate as described below.

The *vectorize* step converts the raster layer from the raster representation to a vector representation. It converts each pixel to a point where the location of the point is determined using the ($\mathcal{G}2\mathcal{W}$) mapping. Each point is associated with the corresponding value from the raster layer, e.g., temperature. The *spatial join* step runs an inner-join operation between the set of polygons and the set of vectorized points to produce polygon-point pairs for each point that lies inside a polygon. Finally, the *grouped aggregate* step groups all pairs by polygon ID and runs the user-defined accumulator on the values in each group to produce the final result.

The main drawback of this method is that as the resolution of the raster layer increases, more points will be created in the vectorization step, hence, more point-in-polygon tests will occur on the spatial join step. With hundreds of billions of pixels, this method takes excessively long time as we show in Section 4.

2.3 Rasterization Method

Similar to the big vector data systems, there were several systems that were designed to process big raster data such as SciDB [13], RasDaMan [2], and GeoTrellis [11]. These systems can run the zonal statistics problem in three steps, rasterize, clip, and aggregate, as described below.

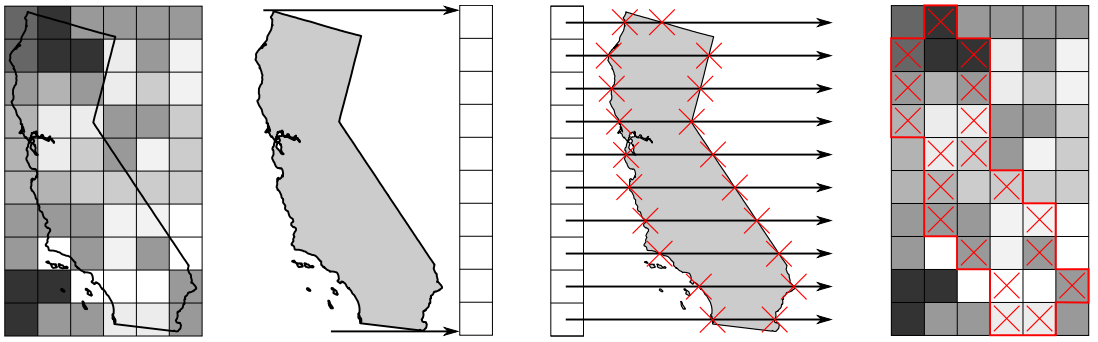
The *rasterize* step uses the traditional polygon filling algorithm to produce a new raster layer, called the *mask*, which has the same size of the input raster layer, called the *data* layer. In the mask layer, pixels that are inside the polygon have values of *one* and other pixels have the value *zero*. The *clip* step overlays the mask and data layers, which have the same size, and removes all pixels in the data layer that have corresponding *zeros* the mask layer by giving them a special marker value, e.g., -1. Finally, the *aggregate* step computes the desired aggregate function, e.g., average, on the retained (non-removed) pixels. These steps are repeated for each polygon in the vector layer.

The main drawback of the above algorithm is that the size of the mask layer grows massively for high-resolution data which requires extra storage and processing.

3 SCANLINE METHOD

In this section, we propose a new algorithm for the zonal statistics problem called the *scanline* method. Unlike the baseline methods described in Section 2, this method processes the two inputs in their raw format and does not require an explicit conversion from raster to vector or vice-versa.

Figure 2 shows a high-level overview of the scanline method which runs in three steps. In step 1, the polygon boundaries are scanned once to locate the lowest and highest points in the polygon, e.g., the two points with the minimum and maximum latitudes. These two points are mapped to the raster layer using the $\mathcal{W}2\mathcal{G}$ mapping to locate the range of rows to process in the raster layer.



(a) Input (b) Step 1: Locate first and last row (c) Step 2: Find intersections with scan lines (d) Step 3: Process the pixels

Figure 2: Scanline method

Step 2 runs several scanlines from the center of each pixel and finds the intersections of each scanline with the polygon boundary. More specifically, for each row in the range computed in step 1, it uses the $\mathcal{G}2\mathcal{W}$ mapping to find the corresponding y-coordinate in the vector space. For each y-coordinate, it computes the intersection between the horizontal scanline at that y-coordinate and each segment on the polygon. The intersections in each row are then sorted by their x coordinates.

Step 3 maps each intersection to a pixel in the raster layer. Then, the raster layer is processed row-by-row by processing every consecutive range of pixels between two intersections in each row. This is the only step that actually requires access to the values in the matrix of the raster layer. This means that the scanline method only reads and processes the pixels that lie inside the polygon which makes it optimal in terms of disk IO.

Notice that this algorithm assumes that all pixels in each row in the raster layer map to a horizontal line, i.e., scanline, in the vector layer. This property holds if both the vector and raster layers belong to the same coordinate reference system (CRS). We guarantee this by projecting the polygon to the same coordinate reference system of the raster layer.

This algorithm overcomes the limitation of the two baseline methods described in Section 2. First, it only requires a minimal intermediate storage for the intersection points. Even these intersections can be done row-by-row and we do not have to store all intersections of all rows at the same time. Second, it only accesses the pixels that are inside the polygon which improves disk IO for very large raster layers. Third, it does not require any complicated point-in-polygon tests which makes it much faster than the vectorization method. As we will show in Section 4 this method is IO-bound which makes it optimal for the processing perspective.

4 EXPERIMENTS

This section provides an exhaustive experimental evaluation using real data to show the efficiency of the proposed algorithms. Section 4.1 describes the setup of the experiments while Section 4.2 provides the results.

Table 1: Vector and Raster Datasets

Vector datasets

Dataset	Polygons	Segments	$\frac{\#segments}{\#polygons}$	File Size
Counties	3,108	51,638	17	978 KB
States	49	165,186	3,370	2.6 MB
World	284	3,817,412	13,440	60 MB

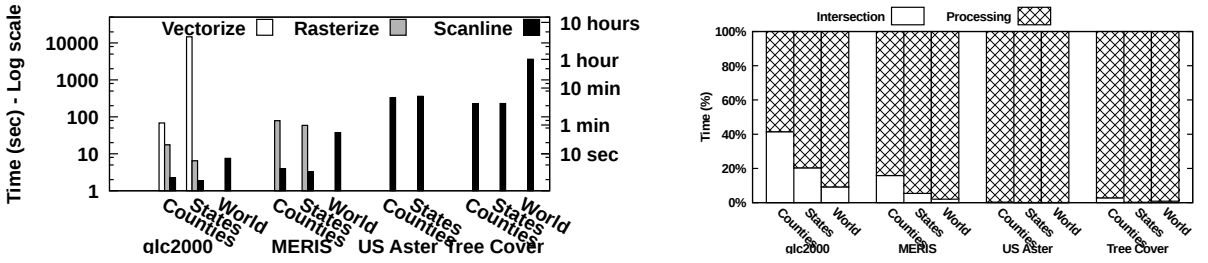
Raster datasets

Dataset	Resolution in Pixels	File Size
glc2000	40,320×16,353	629 MB
MERIS	129,600×64,800	7.8 GB
US-Aster	208,136×89,662	35 GB
Tree cover	1,296,036×648,018	782 GB

4.1 Setup

We run all the experiments on a single machine with Intel Xeon E3-1220 v5 3.00GHz quadcore processor, 64 GB of RAM, and a 2 TB HDD running Ubuntu 16.04.2 and Oracle Java 1.8.0_102. The methods are implemented using the open source Geotools library 17.0. The rasterization method is implemented using PostgreSQL 9.3 and PostGIS 2.3.2. In all the techniques, we compute the four aggregate values, minimum, maximum, sum, and count. We measure the end-to-end running time as the performance metric which includes reading both datasets from disk and producing the final answer.

Table 1 lists the datasets that used in the experiments. The vector layers represent the US continental counties and US continental states with 3000 and 49 features respectively. The Large-Scale International Boundaries (LSIB) includes geographic national boundaries for 249 countries and disputed areas. The raster datasets come from various government agencies. The GLC2000 and MERIS 2005 datasets are from the European Space Agency with pixel resolutions of 0.0089 decimal degrees (1km) 0.0027 (300m) respectively. The US Aster dataset originates from the Shuttle Radar Topography Mission (SRTM) and covers the continental US. Hansen developed the global Tree Cover change dataset which covers the entire globe. Both datasets have a spatial resolution of 0.00028 decimal degrees (30m).



(a) Overall running time

(b) Time breakdown

Figure 3: Experimental results

4.2 Experimental Results

Figure 3(a) gives the overall comparison between the two baseline methods and the proposed scanline method. Scanline is several orders of magnitude faster than the baselines. Furthermore, it was able to process the high-resolution datasets, US Aster and Tree Cover, where the baseline methods failed. Notice that the performance of scanline is almost the same with US counties and states which both cover the same pixels in the continental US. The reason is that the running time of the scanline algorithm is dominated by disk IO while the CPU overhead is negligible. This makes the scanline method optimal for the case where the raster file has to be loaded from disk. The running time can still be improved in the future using the following two methods. (1) Preprocessing the raster file to provide partial aggregates for big chunks that can minimize disk IO as was done earlier in [4]. (2) Running in a distributed environment where multiple disks can be accessed simultaneously.

Figure 3(b) shows the breakdown of the scanline method running time into two phases, intersection and processing. The intersection phase contains all the processing shown in Figure 2 except for reading the pixels from the raster file. The processing phase is the one that reads the pixel values and accumulates them. All numbers are normalized to the overall running time for readability. This experiment shows that as the size of the raster layer increases, the processing phase dominates the running time. This makes the scanline approach near optimal as it is dominated by disk IO for reading only the pixels that need to be processed.

5 CONCLUSION

In this paper, we discussed the zonal statistics problem which combines both vector and raster datasets. We showed that the two baseline methods that convert the raster to vector or the vector to raster layers provide a subpar performance due to the massive amount of intermediate data. Alternatively, we proposed a novel method, termed scanline, which achieves orders of magnitude speedup by processing the two inputs in their raw format without the need of an intermediate representation. This paper is just a proof of concept for the combination of vector and raster datasets in the query processor and researchers can follow up with similar algorithms for other problems.

REFERENCES

[1] Ablimit Aji, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, and Joel H. Saltz. 2013. Hadoop-GIS: A High Performance Spatial Data Warehousing System over MapReduce. *PVLDB* 6, 11 (2013), 1009–1020.

[2] Peter Baumann, Andreas Dehmel, Paula Furtado, Roland Ritsch, and Norbert Widmann. 1998. The Multidimensional Database System RasDaMan. In *SIGMOD*. Seattle, WA, 575–577.

[3] Ahmed Eldawy and Mohamed F. Mokbel. 2015. SpatialHadoop: A MapReduce Framework for Spatial Data. In *ICDE*. Seoul, South Korea, 1352–1363.

[4] Ahmed Eldawy, Mohamed F. Mokbel, Saif Alharthi, Abdulhadi Alzaidy, Kareem Tarek, and Sohaib Ghani. 2015. SHAHED: A MapReduce-based System for Querying and Visualizing Spatio-temporal Satellite Data. In *ICDE*. Seoul, Korea, 1585–1596.

[5] EOSDIS 2017. The Common Metadata Repository: The Foundation of NASA's Earth Observation Data. (2017). https://earthdata.nasa.gov/the-common-metadata-repository.

[6] ESA 2017. The ESA Earth Observation Payload Data Long Term Storage Activities. (2017). https://www.cosmos.esa.int/documents/946106/991257/13_Pinna-Ferrante_ESALongTermStorageActivities.pdf/813babe0-58db-4e23-b710-3bd9d6b58b12.

[7] David Haynes, Steven Manson, and Eric Shook. 2017. Terra Populus' Architecture for Integrated Big Gepspatial Services. *Transactions on GIS* (2017).

[8] David Haynes, Suprio Ray, Steven M. Manson, and Ankit Soni. 2015. High Performance Analysis of Big Spatial Data. In *Big Data*. Santa Clara, CA, 1953–1957.

[9] G Darrel Jenerette, Sharon L Harlan, Anthony Brazel, Nancy Jones, Larissa Larsen, and William L Stefanov. 2007. Regional Relationships Between Surface Temperature, Vegetation, and Human Settlement in a Rapidly Urbanizing Ecosystem. *Landscape Ecology* 22 (2007), 353–365. Issue 3.

[10] G. Darrel Jenerette, Sharon L. Harlan, William L. Stefanov, and Chris A. Martin. 2011. Ecosystem Services and Urban Heat Riskscape Moderation: Water, Green Spaces, and Social Inequality in Phoenix, USA. *Ecological Applications* 21 (2011), 2637–2651. Issue 7.

[11] Ameet Kini and Rob Emanuele. 2014. Geotrellis: Adding Geospatial Capabilities to Spark. (2014).

[12] Shoji Nishimura, Sudipto Das, Divyakant Agrawal, and Amr El Abbadi. 2013. MD-HBase: Design and Implementation of an Elastic Data Infrastructure for Cloud-scale Location Services. *DAPD* 31, 2 (2013), 289–319.

[13] Michael Stonebraker, Paul Brown, Donghui Zhang, and Jacek Becla. 2013. SciDB: A Database Management System for Applications with Complex Analytics. *Computing in Science and Engineering* 15, 3 (2013), 54–62.

[14] Randall T. Whitman, Michael B. Park, Sarah A. Ambrose, and Erik G. Hoel. 2014. Spatial Indexing and Analytics on Hadoop. In *SIGSPATIAL*. Dallas, TX, 73–82.

[15] XFEL 2017. European XFEL: Data Handling. (2017). http://www.xfel.eu/research/data_handling/.

[16] Dong Xie, Feifei Li, Bin Yao, Gefei Li, Liang Zhou, and Minyi Guo. 2016. Simba: Efficient In-Memory Spatial Analytics. In *SIGMOD*.

[17] Jia Yu, Mohamed Sarwat, and Jinxuan Wu. 2015. GeoSpark: A Cluster Computing Framework for Processing Large-Scale Spatial Data. In *SIGSPATIAL*. Seattle, WA, 70:1–70:4.