# High Performance Analysis of Big Spatial Data

David Haynes[1], Suprio Ray[2], Steven M. Manson[3], Ankit Soni[1]
[1]Minnesota Population Center, University of Minnesota
{dahaynes, asoni}@umn.edu
[2]Faculty of Computer Science, University of New Brunswick, Fredericton
sray@unb.ca
[3]Department of Geography, University of Minnesota
manson@umn.edu

***Abstract*— Every year research institutions produce petabytes of data. Yet, only a small percent of the data is readily accessible for analysis. Terra Populus acts as the bridge between big data sources and researchers. Researchers are provided convenient web applications that allow them to access, analyze, and tabulate different datasets under a common platform. Terra Populus is developing three unique applications. The first application, Paragon, is a prototype parallel spatial database, which aims to extend the functionality of PostgreSQL and PostGIS onto multinode systems. Terra Populus' Tabulator application employs Parquet on Spark to build dynamic queries for analyzing large population survey data. The last application, Terra Explorer, is an exploratory analysis tool for visualizing the spatial datasets within the repository.**

***Index Terms*— big spatial data, spatial analysis, spark, parquet, web mapping, dynamic visualization**

## I. Introduction

The Terra Populus project demonstrates emerging capabilities in integrating complex big data – namely, heterogeneous spatial data – within high performance computation environments. Terra Populus identifies, acquires, and develops various data sources ranging from historic handwritten census forms to current satellite observations of the earth. The aim of the project is to preserve these data and make them Internet-accessible, so they are readily available for scholars, students, policy makers, and members of the public.

Terra Populus is a member of the National Science Foundation, Datanet Initiative, the goal of which is to develop cyber infrastructure that preserves, integrates, and provides open access to scientific data. Terra Populus utilizes an open source ecosystem to provide access to heterogeneous big data (microdata, raster data, and vector data) for researchers studying a range of social and natural systems along with human-environment interactions. A primary challenge for this project has been developing methods that preserve the integrity of the underlying data, are sufficiently flexible to support the needs of many different users, and scale for high performance computation. The tools and platforms that we are developing provide users with the ability to create tailored queries, that may be analyzed quickly, which in turn fosters the growth of scientific knowledge.

The Terra Populus data collection exemplifies two of primary "V's" commonly attributed to big data: variety, volume. We deal with data from a wide array of sources (variety); some of the large databases in the world (volume); and with remotely sensed imagery, among others, which are collected in large and rapidly increasing amounts. Terra Populus also deals with big data characterized by two newer "V's", namely value (integrating our data products provides new greater scientific understanding) and veracity (our data collection is composed of gold-standard which is vetted for research).

The data processing toolset that we are developing seeks to accommodate all of these big data characteristics. There are a growing number of big data processing and analytics toolsets, yet there are is a paucity of tools (or even basic research) that work with heterogeneous big spatial data or provide interoperability of between datasets. This paper describes three computational challenges that exemplify 'big spatial data' and are being tackled by the Terra Populus project. We examine in particular our approaches for high performance spatial analysis, dynamic tabulation, and dynamic visualization.

## II. High Performance Spatial Analysis: Paragon

Spatial databases are the driving force behind most traditional GIS applications, in that, domains ranging from land surveys to city planning or resource monitoring depend on the existence of a spatial database on which all other operations rely. While big data applications are remaking the form of spatial databases, particularly due to the rapid rise in data volume, there will always remain a need for spatial analysis. As a result, we are seeing new classes of spatial applications and Terra Populus is developing new methods that operate on spatial data.

Regardless of data source, spatial queries lie at the heart of spatial analysis. Spatial queries can take many forms, including range queries (e.g., return all points of land higher than 4,000m inside a national park), *K* nearest neighbors queries, and join queries (e.g., all rivers that have bridges that cross them), among other. Whereas geospatial Web services such as Google Maps, are driven by short-running range queries, many of the emerging spatial analytics applications are characterized by long-running spatial join queries [6]. For example, the polyline crosses polylines spatial join query involving 73 million records takes over 20 hours [4]. These queries can benefit from high performance query processing. With the growing popularity of the open source MapReduce framework Hadoop, a number of MapReduce based spatial query processing systems have been developed. These include academic projects like Spatial Hadoop [2] and Hadoop-GIS [9], along with commercial projects such as GeoTrellis [5]. However, the set of spatial query features supported by these solutions are still limited, especially compared to standard GIS products that operate on relational tables. This is why Terra Populus is currently exploring other options, including the development of our own system.

Like several other MapReduce based non-spatial systems, there is a trend towards implementing relational database features. This move toward relational-like features has led to the observation [7] that these SQL on MapReduce systems have come full circle, in that they resemble shared-nothing parallel relational databases. Paulson et al. [10] argues that parallel relational databases take advantage of over 20 years of research, which MapReduce systems cannot match. With extensive evaluation, they demonstrated that shared-nothing parallel relational databases perform significantly better than MapReduce systems. Therefore, we have adopted the view that a shared-nothing parallel database can offer the best high performance platform for spatial query processing. Since there is no existing open-source shared-nothing parallel spatial database, we are developing one called Paragon.

Paragon is a parallel spatial database that runs a PostgreSQL database instance in each of the nodes in a cluster of machines. Spatial support was added to PostgreSQL in 2005, as part of PostGIS extension. While the initial release only included support of vector data sets, subsequent versions have included support for raster datasets. As PostGIS has evolved over the years, so have parallel environments for PostgreSQL. When first developed it only operated on a single core; even today when placed on servers with multiple cores it can only utilize a single core per query. Multiple projects have made an effort to scale PostgreSQL. GridSQL was one of the first notable projects to make such an effort, followed by Stado [12]. Stado supports vector datatypes and but lacks full functionality for distributing spatial data across a cluster of machines.

There are a number of research issues that are pertinent to Paragon in the context of heterogeneous big spatial data management. In a shared-nothing parallel database, the dataset is horizontally partitioned and assigned to each node in the cluster. With spatial data, this partitioning process is known as spatial declustering. This process involves logically decomposing the spatial domain into 2-dimensional partitions. These partitions are called “tiles”. The spatial domain could be partitioned into regular grids or using another method such as recursive decomposition based on a constraint on the object count in each tile. Since a spatial object’s extent may overlap more than one of these tiles, it may be necessary to replicate such an object to every partition that it overlaps. Due to these considerations, there exists a few design choices and as a consequence several spatial declustering approaches have been proposed [18, 19, 4, 11]. The creation of tiles through spatial declustering enables parallel join, as each tile could be processed independently. The overall performance of a spatial join is determined by the execution time of the slowest tile. The spatial declustering approach must take into account of the object distribution skew, processing skew and the fact that the refinement step dominates the two step spatial query evaluation [11]. Furthermore, within the Terra Populus project, a key requirement is to be able to join vector and raster datasets. Therefore, the spatial declustering approach must address the fact that vector datasets are discrete single layer objects, whereas rasters can be comprised of multiple layers with varying spatial resolutions. Paragon supports both vector and raster data types. We are developing spatial declustering algorithms that address the above mentioned issues.

Additionally, the query scheduler needs to coordinate query tasks intelligently among computing nodes. Paragon executes a spatial join query in an iterative fashion, each node processing one partition at a time. This is inspired by our previous work [4, 11]. However a limitation of the previous approach is that each node needed to host the entire set of partitions. We are exploring spatial partitioning approaches, such that each node can host a subset of the partitions. Finally, to achieve good performance, the query optimizer needs to generate “ideal query plans” [4] while executing spatial join queries, essentially taking spatial locality into account. Although Stado supports vector-based spatial queries, it does not support spatial declustering based data distribution and does not generate ideal query plans with spatial queries. As a result, the parallel performance of a spatial join query with Stado could actually be worse than that of the sequential execution with a PostgreSQL instance. For example, we demonstrated in [20] that the execution time of the query “Polyline Intersects Polygon” with a 2-node Stado system is significantly longer than that with a single instance PostgreSQL.

To address the above-mentioned issues we have developed Paragon, as an extension to Stado [12]. Paragon exploits a cluster of nodes, each of which hosts a PostgreSQL/PostGIS database instance. We present a preliminary evaluation of Paragon for a dataset consisting of line and polygon objects from the TIGER [6] California dataset. The details of the dataset are shown in table 1.

The current implementation of Paragon uses a variant of round-robin declustering. The declustering algorithm produced 1024 spatial partitions after processing the dataset in Table 1. The physical storage and management of the partitions in Paragon is done by taking advantage of PostgreSQL’s sharding feature [16]. We extended the SQL create table statement to specify spatial declustering parameters, such as, the number of partitions to be created, the declustering method, and a label for the declustering scheme. To execute a spatial join, the labels of the two tables, being joined, must match. This mechanism allows the same spatial dataset to be partitioned using different declustering schemes.

Table 1. Spatial data used for comparison

| Database Table (acronym) | Geometry | Number of Objects |
|---|---|---|
| Area-water (Aw) | Polygon | 39,334 |
| Area-landmass (AI) | Polygon | 5,5951 |
| Edge (Ed) | Polyline | 4,173,498 |

Table 2. Comparison of Query Times: Paragon vs PostgreSQL

| Query (acronym) | PostgreSQL (seconds) | Paragon (seconds) | Speedup |
|---|---|---|---|
| Polygon overlaps Polygon (Aw_ov_Aw) | 77.3 | 53.5 | 1.37 |
| Polyline Touches Polygon (ED_to_Al) | 452.9 | 246.0 | 1.84 |
| Polyline Crosses Polyline (Ed_cr_Ed) | 1693.2 | 1022.0 | 1.65 |

We executed spatial join queries from the Jackpine spatial database benchmark [6] with Paragon in a two node cluster. The queries are expressed in SQL with some of the spatial predicates adopted by Open Geospatial Consortium (OGC). For instance, Code 1 demonstrates the "Polyline Touches Polygon" query shown in Table 2.

Code 1. Spatial SQL Query

```
SELECT COUNT(*) FROM edges ed, arealm al WHERE
ST_Touches(ed.geom, al.geom);
```

A comparison of observed single run execution times is shown in Table 2. The results show, Paragon achieves near linear speedup with 2 nodes, which bodes well for moving the system to multiple nodes. We plan to evaluate Paragon with a larger cluster in the future.

Although optimized for spatial join, since Paragon is a parallel relational database, it is well suited for complex joins involving spatial and non-spatial tables. Our hope is that Paragon will become a useful tool for big geospatial data processing to the research community as well as enterprises

## III. Dynamic Tabulation

Microdata is the term used to denote survey data on individuals and households collected by government census agencies. It is extremely rich data, due to its flexibility for tabulation. Since a microdata record contains all the responses of a single individual, researchers are able to create new tabulations from the original data. Currently Terra Populus is investigating high performance computation platforms that will allow users to build their own tabulations from microdata datasets and its variables. A Terra Populus user can use the system and tabulate the data by selecting the datasets, the row and column variables and any constraints or filters which should be applied on those variables. Usually a geographic variable (e.g. county code) forms a row variable, in order for users to analyze data geographically, however that is not a necessary.

For example, a researcher may be interested in studying the effect of education attainment on marital status and conception of first child. The proposed system could generate/tabulate this targeted population data-set to the researcher through a tabulation definition, where the sex is female, who have had at least one child, age is between 16-45, and tabulated against all levels of education. Such tabulations are common in social science research and support specific research questions. In this example, the system would construct a spark operation similar to, Code 2.

Code 2. Pseudo Code in Python

```
data_frame.filter("SEX = F AND NUMBER OF
CHILDREN>1 AND 16<=AGE<=45").groupBy(["dataset",
"AGE", "EDUCATION ATTAINMENT", "MARITALSTATUS",
"ELDEST CHILD"]).count()
```

In the initial design, we developed a tabulator which had pre-defined rules on each variable/column of each data-set. In this case, users were only allowed to selected data defined by those pre-defined rules. For example, the variable AGE was divided into subgroups of 0 to 4, 5 to 9 and so on. The data was physically limited to theses predefined aggregations. Therefore, when using the initial tabulator aggregations could only take place on the defined variable/columns rules like age_5_9. However, given the size and granularity of different data-sets, we proposed designing a new tabulator that removes all pre-existing aggregations, placing all the age data under a single variable/column AGE. The result is that the end-user is has more functionality at their disposal and can create custom queries that are appropriate for the study, such as tabulating all persons age six (e.g. age= 6). This increases the user querying flexibility on our system and provides ability to define custom queries as described in Code 2.

During our initial phase tests, we also found that relational databases do not scale well when working with data-sets where the table structure is in a de-normalized form with a large number of columns. This is due to the nature of traditional databases, where they carry most columns of the rows while performing operations on the selected data. This makes on the fly operations difficult to perform when the number of columns becomes large. In addition, we have evidence that aggregation or large ranged queries do not also perform well on these de-normalized form tables. This led us towards column store databases.

The proposed tabulator employs a columnar storage database due to the reasons that storage, retrieval, selection and filter operations can be done efficiently. As our data extraction system utilizes PostgreSQL and PostGIS for spatial data interoperability, we investigated using a columnar storage extension for PostgreSQL, cstore_fdw, which is a foreign data wrapper for PostgresSQL. However, we have moved away from using cstore_fdw for tabulation as it has an upper limit of 1600 columns, whereas some data-sets or union of data-sets can contain more columns than that.

Currently we are testing Apache Spark's Parquet as the columnar storage database [13] for our next generation tabulator. Parquet allows for columnar storage in which columns are type-casted to Parquet's inbuilt data types, allowing for greater compression per data type. This approach gives a high compression ratio while still allowing for fast data fetching. This is based on an existing record shredding and assembly algorithm [14]. Although converting regular flat files to Parquet format is a time consuming process, it does make up for it by optimizing the query performance on large amounts of data. Since our usecase has rare updates, a single Parquet conversion results in better query performance over time. Once converted to Parquet format, the datasets can then be queried via Spark SQL [15].

We tested the performance of Parquet using IPUMS-I [17] datasets, which is the primary microdata source of Terra Populus[1]. These tests were performed on a single machine with eight processors and six gigabytes of memory. Table 3 shows multiple query results. Query 1 returns a summary count of records, where the records are filtered by a column value and grouped by data (e.g. count of males in Argentina 1970).

Code 3. Spark SQL Query 1 in Python

```
all_people.filter("SEX = 1").groupBy(["SEX",
            "dataset"]).count()
```

Query 2 employs filters on two columns, and performs three aggregations (e.g count of males 15-20 years old in Argentina 1970). Query 3 employs four filters on three columns and performs aggregations on five column variables. The dataset chosen for the single dataset test, row 1, was Brazil 2010 which has 309 columns. The datasets chosen for row four consists of multiple country datasets in South American. The dataset has 4326 unique columns across 56 different datasets. The datasets chosen for rows two and three are subsets of the data found in row four.

Table 3. Parquet query performance on multiple datasets

| Number of datasets | Number of records | Time for Query 1 | Time for Query 2 | Time for Query 3 |
|---|---|---|---|---|
| 1 | 12 million | 5 seconds | 9 seconds | 10 seconds |
| 10 | 25 million | 7 seconds | 10 seconds | 18 seconds |
| 20 | 82 million | 10 seconds | 12 seconds | 27 seconds |
| 56 | 128 million | 7.5 seconds | 20 seconds | 30 seconds |

## IV. Dynamic Visualization

The integration and development of our data repository will grant researchers access to new datasets and formats. While many of our existing users are used to, and prefer, text-based interfaces, we expect some portion of our user community to want visual/spatial interfaces as well. We are therefore developing a web mapping application that acts as a data exploration tool. Terra Populus is rare among human-environment data portals in that users have the ability to tailor their extracts to the datasets they need. This presents opportunities and challenges because users may identify or exclude datasets from their extracts. The visualization application that we are developing helps users understand the availability of data within the study area of their choice.

The application has three main components: a web mapping pane, data selection pane, and a data visualization pane, Fig. 1. We use Geoserver, a Java server application that provides spatial data delivery for web applications [8]. The application uses the JavaScript library OpenLayers to consume and display these spatial datasets on the mapping interface. Through this application Terra Populus integrates spatial and non-spatial datasets.

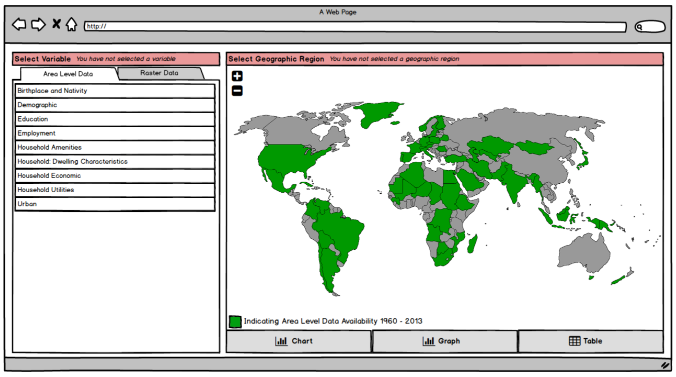
A Web Page
http://
Select Variable You have not selected a variable
Area Level Data
Raster Data
Birthplace and Nativity
Demographic
Education
Employment
Household Amenities
Household: Dwelling Characteristics
Household Economic
Household Utilities
Urban
Select Geographic Region You have not selected a geographic region
🟩 Indicating Area Level Data Availability 1960 - 2013
Chart
Graph
Table

Figure 1. Terra Explorer Landing Page

The primary purpose of the application is to support data exploration, and our development efforts have therefore focused on implementing a dynamic and interactive framework that reflects the other applications within our eco-system. Currently, the application allows users to interactively visualize vector and raster datasets. Areal datasets are dynamically created and styled using choropleth mapping techniques and Geoserver's REST SLD API. Variable substation is applied to rasters to elicit a similar effect. In this approach we each value of a categorical raster is an independent category. For example, IGBP MODIS satellite imagery has pixel values which range from 0 to 255, with pixel values 0-16 representing landcover types (i.e. 0 = water, 12 = croplands). The application allows the user to adjust the opacity values for any of raster pixel value. This enhances the user's understanding of the presence of particular components of dataset within their study region.

The web mapping interface allows for data exploration, and facilitates the understanding of presence of data. However, web mapping is not the only visualization tool in our application. Data visualizations (i.e. bar charts and plots) combined with mapping can enhance the user's understanding of data. Data summarization of our areal (population demographic) and tabulated datasets are relatively straightforward and can be implemented using the previously described tabulator. However, real-time summarization of raster datasets is a computationally intensive problem that remains unsolved. To implement this capacity we have worked with the Spatial Hadoop team to develop a prototype spatial overlay engine [2]. This new prototype supports zonal statistic calculations of rasters in their native GeoTIFF format using Spark.

The prototype engine calculates zonal statistics for any spatial region, and returns via API summary statistics (i.e. min, max, mean, sum, count). To perform this analysis a spatial overlay between vector and raster datasets, is implemented with the Spatial Hadoop platform. The engine must be able to handle different spatial projections, as the project utilizes datasets derived from different satellite instruments. To reduce data errors caused from projections, we project vector datasets to the raster coordinate system and then perform the zonal analysis. The first step is to tile and compress the raster. This reduces I/O costs and memory challenges. While doing this, we create a metadata file that provides the spatial bounding box of each compressed raster tiles. Next a scan-line fill algorithm is used to determine, which pixels lie within the geographic unit. The third step aggregates the pixels and their values by zone/geographic unit and returns the result. The resulting data can then be visualized with data visualization panes.

## V. Conclusion

Terra Populus is advancing our ability to integrate complex big data in a high performance computation environment. Terra Populus has chosen to employ open source software to manage a large range of data in a mix of formats (microdata, raster data, and vector data) for use by researchers studying a range of social, biophysical, and human-environment systems. While there are a growing number of open-source candidates to meet the challenges faced by Terra Populus, there is a profound need for a general framework that operates on heterogeneous big data. We focus on the solutions for high performance spatial analysis, dynamic tabulation, and dynamic visualization.

## References

[1] Terra Populus. http://www.terrapop.org/
[2] A. Eldawy, M. F. Mokbel. SpatialHadoop: A MapReduce Framework for Spatial Data. ICDE. 2015.
[3] S. Wang. A CyberGIS Framework for the Synthesis of Cyberinfrastructure, GIS, and Spatial Analysis. AAAG. 2010.
[4] S. Ray, B. Simion, A. D. Brown and R. Johnson. A Parallel Spatial Data Analysis Infrastructure for the Cloud. SIGSPATIAL GIS. 2013.
[5] GeoTrellis. http://geotrellis.io/
[6] S. Ray, B. Simion, and A. D. Brown. Jackpine: A Benchmark to Evaluate Spatial Database Performance. In ICDE. 2011.
[7] A. Floratou, U. M. Minhas and F. Özcan. SQL-on_Hadoop: Full Circle Back to Shared-Nothing Database Architectures. VLDB. 2014.
[8] GeoServer. http://geoserver.org
[9] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang and J. Saltz. Hadoop-GIS: A High Performance Spatial Data Warehouse System Over MapReduce. VLDB. 2013.
[10] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker. A comparison of approaches to large-scale data analysis. SIGMOD. 2009.
[11] S. Ray, B. Simion, A. D. Brown and R. Johnson. Skew-Resistant Parallel In-memory Spatial Join. SSDBM. 2014.
[12] Stado. https://wiki.postgresql.org/wiki/Stado
[13] Parquet. https://parquet.apache.org/
[14] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. 2010. Dremel: interactive analysis of web-scale datasets. VLDB. 2010. 330-339.
[15] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. 2015. Spark SQL: Relational Data Processing in Spark. SIGMOD 2015. 1383-1394.
[16] PostgreSQL Partitioning. http://www.postgresql.org/-docs/8.3/static/ddl-partitioning.html
[17] IPUMS-I. http://www.ipums.org
[18] J. M. Patel and D. J. DeWitt. Partition based spatial-merge join. SIGMOD. 1996.
[19] J. M. Patel and D. J. DeWitt. Clone join and shadow join: two parallel spatial join algorithms. SIGSPATIAL GIS. 2000.
[20] D. Haynes, S. Ray, S. Manson, D. Van Riper, A. Soni and A. D. Brown. Towards A High Performance System for Heterogeneous Big Spatial Data. CyberGIS AHM, 2015.