DOI: 10.1111/tgis.12955

REVIEW ARTICLE

Transactions in GIS WILEY

Interactive analysis of big geospatial data with high-performance computing: A case study of partisan segregation in the United States

Devika Kakkar | Ben Lewis | Wendy Guan

Center for Geographic Analysis, Harvard University, Cambridge, Massachusetts, USA

Correspondence
Devika Kakkar, Center for Geographic Analysis, Harvard University, Cambridge, MA, USA.
Email: kakkar@fas.harvard.edu

Funding information
This work is partially funded by NSF award #1841403 and OmniSci, Inc. Funding was also provided by Dr. Ryan Enos in the Department of Government at Harvard University.

Abstract

Researchers are increasingly working with large geospatial datasets that contain hundreds of millions of records. At this scale, desktop GIS systems typically fall short and so new approaches and methods are needed. The objective of this work is to develop new approaches to interactively analyze large datasets and then to demonstrate the usefulness of those approaches using a case study looking at voter, or partisan segregation. Historically, the measurement of partisan segregation has been limited to comparing large geographic areas such as counties or states because researchers only had access to aggregated data. In this case study, however, we measure partisan segregation down to the individual for 180 million U.S. voters using advanced geospatial data science and high-performance computing. This article discusses interactive method development for big geospatial data analysis including techniques used, solutions developed, and processing time statistics.

1 | INTRODUCTION

Increasingly, GIS workflows need to be able to support the processing and visualization of very large datasets. Massive geospatial datasets have become increasingly common, coming from satellite and ground sensors, commercial transactions, social media, online publications, and so forth (Goodchild, 2016). Desktop GIS systems have limited capacity for dealing with big data challenges. Examples of these challenges include the following: unusual data forms, the processing of streaming data, parallel computing, and interactive visualization (Yue & Jiang, 2014).

This article focuses on developing approaches to interactively analyze geospatial big data and demonstrate with a case study. The case study involved using a detailed voter dataset to calculate partisan segregation down

to the level of the individual for 180 million U.S voters. Due to the large size of the dataset, our major challenge was to develop efficient methods for making partisan segregation calculations possible at this scale. Using a combination of the techniques described in Sections 2 and 3, we were able to develop a novel solution which makes K-Nearest Neighbor (KNN) and partisan segregation calculations faster and more computationally efficient than each of the approaches alone. This solution enabled the researchers to understand partisanship segregation across the United States at the individual level (Brown & Enos, 2021).

The voter information was obtained from L2, a commercial data vendor which works with both major political parties in the United States. The data consisted of anonymized information on 180 Million registered voters in the United States consisting of their home addresses and declared party affiliations, that is, Democrat, Republican, or Independent. The affiliation values are binary, that is, either 1 or 0, and representing true or false, respectively. Table 1 shows a few illustrative data records to demonstrate the input data schema. The home addresses were geocoded to obtain the latitude and longitude of each voter. To protect privacy, we are not allowed to share the raw voter file data. Additionally, for security purposes, the data were stored on a secure server not accessible to the web for processing and analysis.

Using this information, we first determined K-nearest neighbors for every voter and calculated distances between each of them. Thereafter, we combined this distance information with their neighbors' affiliation to construct an index of individual-level measures of partisan exposure as described in the following sections.

2 | KNN CALCULATION METHODS

Nearest neighbor search is a common spatial problem and one which comes up in other disciplines as well, such as data mining and machine learning, except that in other disciplines the unit of measurement is not spatial distance but instead some sort of matrix distance based on features of the data. In this work, we focus on the spatial nature of the problem. The following sections describe some of the traditional ways of calculating KNN on spatial data.

2.1 | Distance and buffer-based KNN

The basic approach for calculating KNN involves determining pairwise distances for each location to every other location, sorting the distances from low to high, and then selecting the top K. This approach has a time complexity of $O(n^2)$ and it works well in the case of smaller datasets, but in the case of datasets with thousands or millions of locations, it quickly becomes too slow to be practical. In our case, with a dataset of 180 million voters and their locations, pairwise distance calculations would require more than a trillion calculations. An enhancement to this basic approach is to perform a box-assisted query, for example, search for nearest neighbors only within a 100-mile buffer radius around an individual. This would improve performance by limiting the number of pairwise distance calculations. However, the buffer radius will vary widely depending on geography, population density, and trying out all possible buffer radii for large datasets is computationally intensive.

TABLE 1 Illustrative (not real) records of voter dataset showing inputs

Voter Id	Address	Democrat	Republican	Independent
001_1	9079 Acacia Court Helena, MT 59601	0	1	0
001_2	600 Virginia Dr. Big Spring, TX 79720	0	0	1
002_1	373 Oklahoma St. San Antonio, TX 78249	1	0	0

2.2 | Geohash-based KNN

Geohash encoding converts latitude and longitude into a set of binary strings and then crosses the two sets of strings bit by bit to generate a new set of binary strings (Zhou et al., 2020), converting two-dimensional spatial queries into one-dimensional string matching. With this advantage, geohash can achieve fast queries with time complexity of O(1).

Geohash can be used to find the K-Nearest Neighbors based on common prefixes, allowing one to find all neighbor geohash ranges for a specific geohash resolution and then query those geohash ranges to get the KNN values. The geohash-based approach, however, suffers from two major disadvantages. First, the precision of the grid is arbitrary. The precision of a geohash is a number between 1 and 12 that specifies the number of characters of the geohash. The longer the string, the higher the spatial resolution. The issue here is similar to the problem with the buffer-based KNN approach because if precision is kept too high, it may result in fewer than required neighbors in a geohash and neighbor geohash range. On the other hand, if precision is kept too low, it may result in too many neighbors and calculations. The appropriate precision would depend on geography, population density and K, similar to buffer-based KNN.

Second, geohash has an edge case limitation. Geohash relies on the Z-order curve for ordering the points, so two close-by points might be visited at very different times. Therefore, there are edge cases where nearby locations do not share the same prefixes, for example, locations close to each other but on opposite sides of the 180-degree meridian will result in geohash codes with no common prefix. Similarly, points close to the North and South poles will have very different geohashes. Though this does not affect our use case because we are focused on the United States, this could be a limitation in other instances.

2.3 | Index-based KNN

Indexing speeds up search by organizing the data into a search tree which can be quickly traversed to find a particular record. Without indexing, any search for a feature would require a "sequential scan" of every record. To handle spatial data efficiently, as required in computer aided design and geo-data applications, a database system needs an index mechanism that will help it retrieve data items quickly according to their spatial locations (Guttman, 1984).

An index can make neighbor searches faster and more efficient in a database. KNN in PostGIS is implemented as a pure index-based nearest neighbor search. By walking up and down the index, the search can find the nearest candidate geometries without using any magical search radius number. R-tree is one of the most commonly used spatial indexes. The following section describes it in more detail.

2.3.1 | R-tree-based index

R-tree is a dynamic index structure for spatial searching (Guttman, 1984). Both PostGIS and Oracle spatial use R-tree for indexing spatial data. "R" in the R-tree stands for rectangles as R-tree breaks the data into rectangles, sub-rectangles, sub-sub rectangles, and so on. The key idea is to group nearby objects and represent them with their Minimum Bounding Rectangle (MBR) in the next higher level of the tree. R-tree is a self-tuning index meaning that it tunes itself automatically depending on data density and object size. Figure 1 describes the underlying concept used by R-tree to store spatial data. Each geometry from G1 to G7 is enclosed by its Minimum Bounding Rectangle (MBR) and arranged in the appropriate level in the tree. For example, G1 and G2 are the highest level of R-tree for this data, followed by G3 and G4 and so on. Each node of the R-tree is placed on a separate disk page. This makes the R-tree particularly useful for applications involving very large databases where the index is too large to fit in main memory (Gavrila, 1994). For big datasets, there is an initial cost of processing data into an index but in most cases the cost is worth paying to gain search speed-ups later. However, the issue with PostGIS R-Tree is that it is a tree built incrementally on spatial data and might not have high spatial coherence of the leaves. A balanced R-tree would provide a more

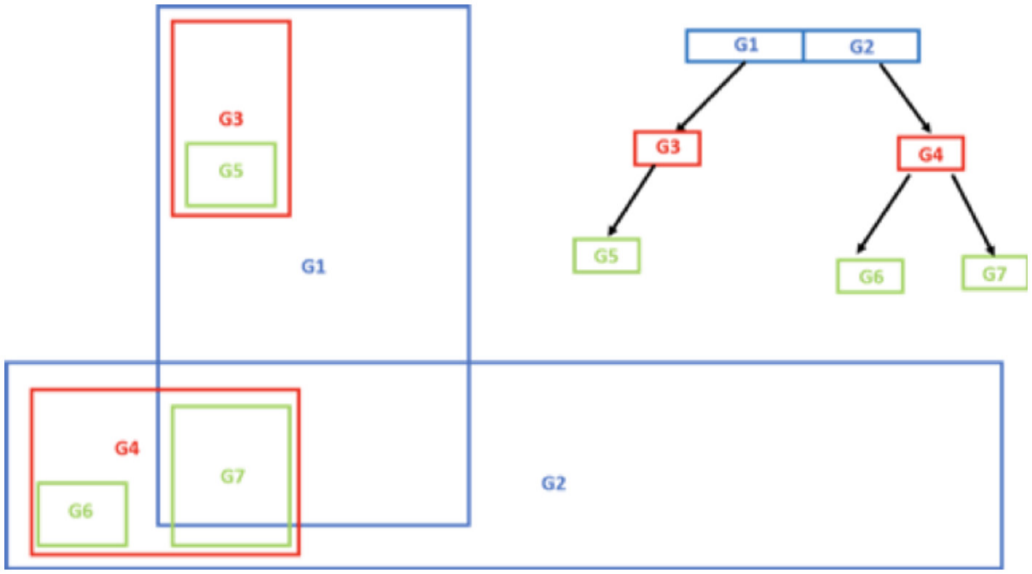

FIGURE 1 R-tree index for spatial data

spatially compact tree and thus further speed up the search. Since there is not a balanced R-Tree algorithm available in PostGIS, we used a geohash proxy that puts spatial data into a spatially autocorrelated order. Clustering on geohash could provide an advantage similar to a balanced R-tree as described in the following section.

3 | GEOHASH-BASED SPATIAL CLUSTERING

Spatial clustering is the process of grouping a set of objects into classes or clusters so that objects within a cluster have high similarity in comparison to one another, but are dissimilar to objects in other clusters (Han et al., 2001). In fields such as GIS, the performance of data access from disk is critical for the performance of applications (Liu et al., 2014). Disk clustering refers to the clustering of data on disk such that similar objects are grouped in similar sections of disk, thereby providing fast and efficient access.

If a dataset is too large to fit in Random Access Memory, which is true in our case, the databases can process information only as fast as it can be found and retrieved from disk. In most cases, data are written to disk without consideration of how the data will be retrieved in the future. Here there is an opportunity to speed up access to huge databases if one can ensure that records which are likely to be retrieved together in the same query result are located in similar physical locations on the hard disk. Since, in our case, we are dealing with spatial data which tends to be accessed in spatially correlated windows, clustering based on a spatial index makes sense.

Figure 2 shows how spatial data would be stored in a clustered vs unclustered manner on disk. Clustering ensures that nearby neighborhoods are in nearby tracks and sectors of the disk, as compared to more random storage in unclustered schema. In geohash-based spatial clustering, locations are clustered based on their geohash and stored on disk with closer points grouped together. This approach provides an advantage similar to a balanced R-tree.

4 | SOLUTION

Building on Sections 2 and 3, this section describes the methodology that was used to analyze the dataset of 180 million voters. Our solution consists of two main processing steps as shown in Figure 3:

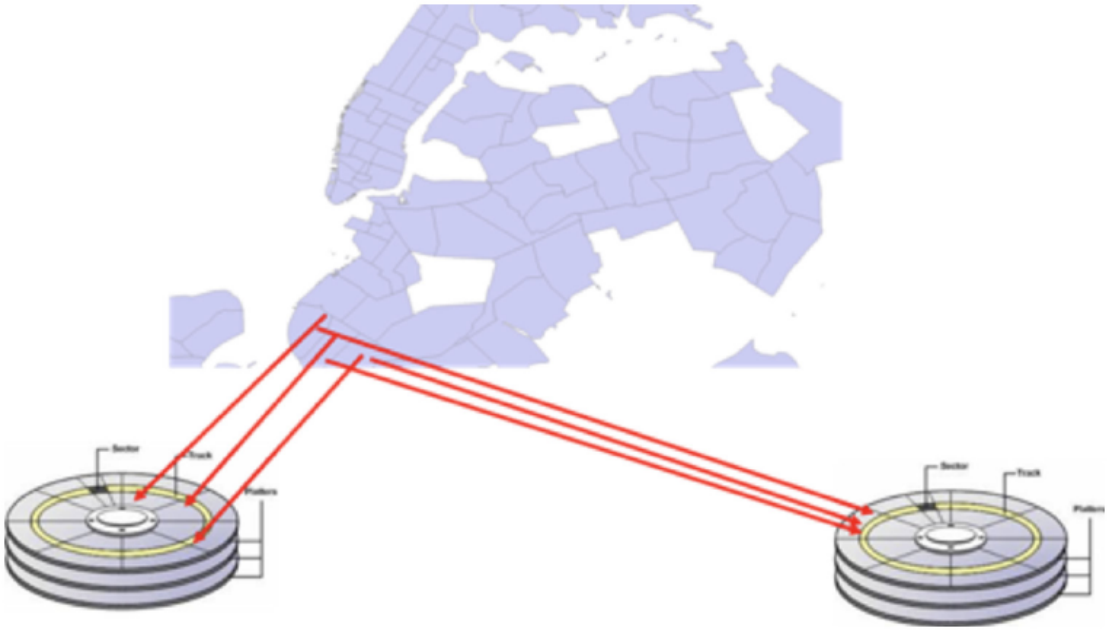

FIGURE 2 Non-clustered (left) versus clustered (right) spatial data storage on disk

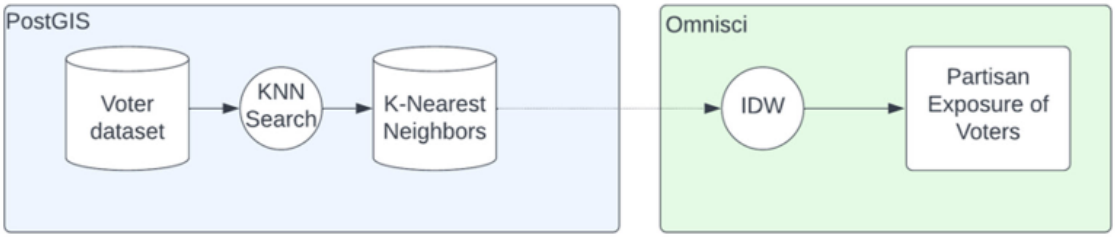

FIGURE 3 Architecture diagram for calculation of partisan exposure index

1. **KNN calculation:** Find K-nearest neighbors for every voter and calculate the Euclidean distance between pairs of voters. This is discussed in Section 4.1.
2. **Partisan Exposure Index calculation:** Calculate a partisan exposure index for each voter using inverse distance weighting (IDW) as discussed in Section 4.2.

4.1 | Computation of K-nearest neighbor for each voter

The two-layered approach we developed combines the powerful techniques of index-based KNN search with appropriate spatial clustering on the disk. First, the voter's home address was geocoded to get the latitude and longitude of each voter. Then, the geohash was calculated for each voter using the latitude and longitude values. Next, spatial clustering was done using these geohash values. Once the data were clustered on the disk, the R-tree-based KNN search in PostGIS was used to find the 1000 nearest neighbors of each voter. Finally, for each of $n = 180$ million geocoded individuals, we measure the distance to their $k = 1000$ nearest neighbors as defined by the closest geodesic distances from the registered voters' residences, creating a distance measure for $n \times k$ (180 billion) relationships. Thus, for every voter, we identify the distance to their 1000 nearest neighbors and combine this information with data on their neighbors' partisanship to construct individual-level measures of partisan exposure index as described in Section 4.2.

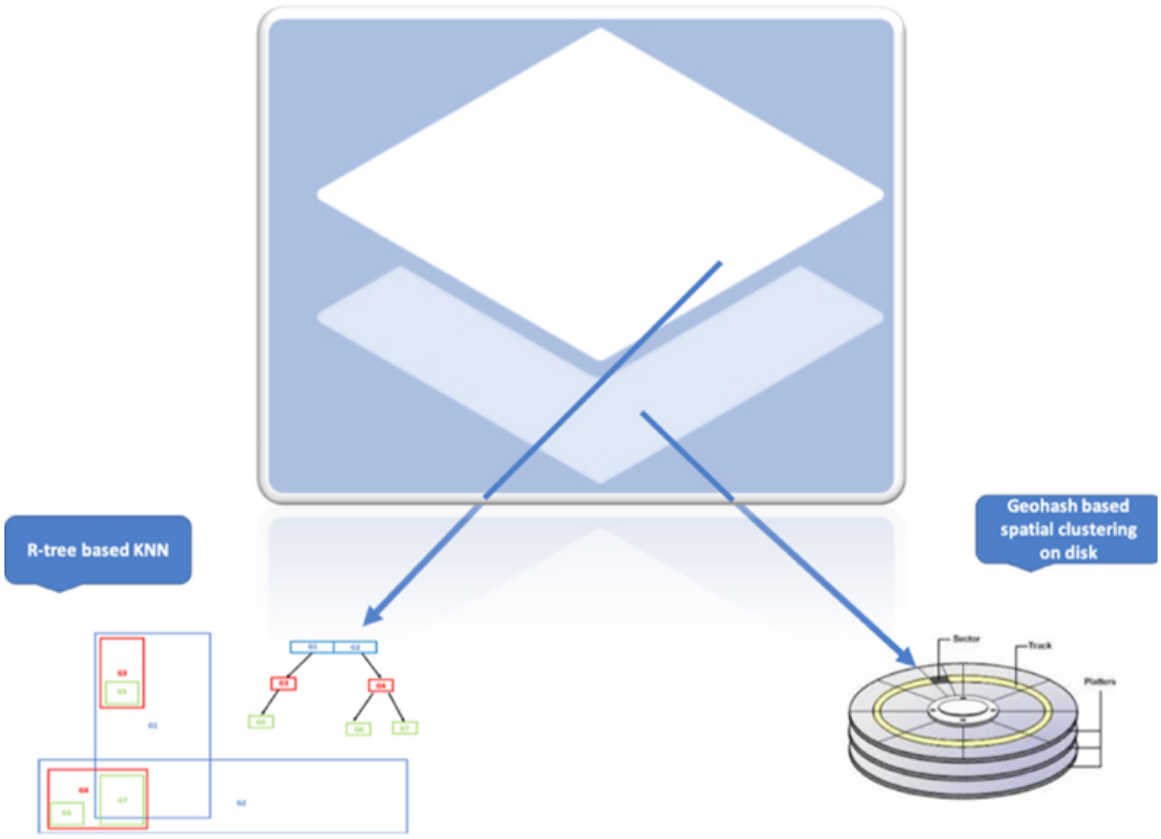

FIGURE 4 A two-layered approach based on geohash-based clustering and R-tree-based KNN

TABLE 2 Comparison of time taken for KNN with different techniques

Method	Time taken (in days)
Sorting by distance	2,628,000 (estimated)
R-tree alone	18
R-tree with Geohash clustering	10

As shown in Figure 4, the bottom layer of geohash-based spatial clustering ensures that records that are likely to be retrieved together in the KNN calculations are located in similar physical locations on the hard disk. This speeds up the calculations by providing fast and efficient access to records on the disk. Furthermore, the top layer of R-tree-based KNN search helps to reduce the number of calculations by searching for a voter's neighbors in its Minimum Bounding Rectangle (MBR) at the same level in the index and incrementing to higher levels until a required number of neighbors are found, for example, 1000. Thus, this approach is significantly faster as it not only reduces the total number of calculations required, but also increases the speed of each calculation. Table 2 compares the time needed to calculate 1000 nearest neighbors for 180 million U.S. voters using various KNN search approaches on a server with 4 vCPU and 16 GB RAM. The time for "sorting by distance" was estimated by running a test for calculating 1000 Nearest Neighbor of 1000 voters using this technique.

4.2 | Computation of a partisan exposure index for each voter

The output from the calculations described in Section 4.1 is a distance measure for 180 billion pairs, that is, distance of each voter (n = 180 million) to their (k = 1000) nearest neighbors. We use these 180 billion distances

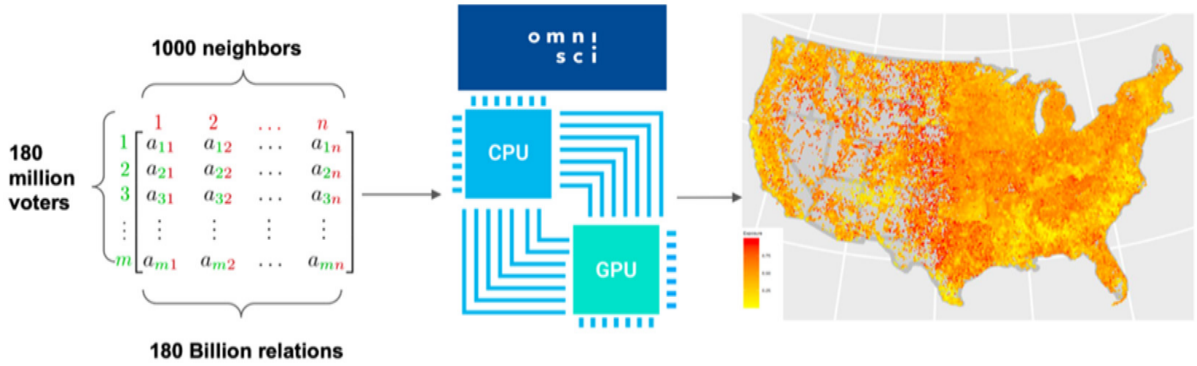

FIGURE 5 Accelerated GPU-based processing of partisan index for every voter

TABLE 3 Comparison of time taken for partisan index calculation with different methods

Method	Time taken (in hours)
R based	200
OmniSci based	17

along with the voter affiliation information to calculate a partisan exposure index for each voter as shown in Figure 5 using the equation below:

$$\rho_i^j = \frac{\sum_{k=1}^{K} \frac{1}{(d_{ik}+c)^a} \times affl_k^j}{\sum_{k=1}^{K} \frac{1}{(d_{ik}+c)^a}} \quad (1)$$

where i is the subject voter for which exposure needs to calculated; j is the partisanship, that is, democrat, independent, and republican; k is the kth neighbor; K is the total number of nearest neighbors used in calculation, for example, 1000; ρ_i^j is the partisan exposure index of ith voter for jth partisanship; $affl_k^j$ is the affiliation score of kth neighbor to jth party, either 0 or 1. Value is 1 in the case of an affiliation and 0 otherwise; d_{ik} is the Euclidean distance between ith subject voter and their kth nearest neighbor, calculated using KNN; c is a constant adjustment to distance weighting; a is an exponent parameter to control the impact of inverse distance weighting.

Initially, the processing of the partisan index stated in (1) was performed on an Amazon Web Services (AWS) r5.xlarge instance with 4 vCPUs and 32 GB RAM, calibrated for parallel processing in R. The total computation time for this was more than 200 hours, and was completed in multiple installments of data chunks (Brown & Enos, 2021). This time cost is huge even without performing any tuning of the model parameters. However, in most cases, it is highly beneficial for researchers to try different models or perhaps tune the parameters of models to find the best fit. Hence, the speed of analytics becomes critical, and the challenge becomes reducing the runtime so that a researcher can interactively tune the model.

To enable interactive tuning of the model we used OmniSci (OmniSci Overview, n.d.), a GPU database which supports running query, visualization, and data science workflows over hundreds of millions of records in milliseconds. OmniSci was run on a server with 2 vCPU, 1 Nvidia A100 GPU, and 32 GB RAM. We found that using OmniSci resulted in a significant speed-up over the R-based approach. The total time for processing the partisan weight of 180 million voters in OmniSci was 17 hours, a speed-up of 91.7% over R as shown in Table 3. The platform was used to perform interactive modeling of parameters a, c, and k on (1).

5 | RESULTS AND DISCUSSION

The techniques described in Section 4 enabled the calculation of partisan exposure of every individual voter in the United States for 180 million voters, and made it possible for researchers to visualize partisan segregation across increasingly small geographies (Brown & Enos, 2021). It was determined that high partisan segregation exists across the country, with most voters of both political parties living in partisan bubbles with little exposure to the other party.

The approach described here provides an efficient and affordable way to solve geospatial big data problems particularly those involving K-nearest neighbor search and Inverse Distance Weighting. It is cost and time effective with extremely fast Input/Output speed, for example, using our two-layered approach for KNN calculation, we achieved a speed of 200,000 distance calculations/second. Additionally, with GPU processing in OmniSci, we were able to calculate the partisan index of 180 million voters in 17 h, a 91.7% speed-up over R.

Big Data is changing the ways data are managed and analyzed. By scaling open-source geospatial tools and combining them with new tools such as OmniSci, big data capabilities can be made available to researchers. This article provides an overview of techniques for performing and scaling geospatial operations involving K-Nearest Neighbors and Inverse Distance Weighting on big data. As a next step, we would like to expand our case study using a historical voter dataset which goes back 20 years. This would help the researchers to study changes in voting behavior over time. In addition, we would like to incorporate other datasets such as social media data, to better explain voter behavior. In the future, we would also like to investigate GPU-based tools such as OmniSci for calculating KNN to further improve performance. Finally, it is also possible to allow queries to span multiple GPU hosts to further accelerate analytics, or support even larger datasets.

ACKNOWLEDGMENTS

This work is partially funded by NSF award #1841403 and OmniSci, Inc. Funding was also provided by Dr. Ryan Enos in the Department of Government at Harvard University.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

Research data are not shared.

REFERENCES

Brown, J. R., & Enos, R. D. (2021). The measurement of partisan sorting for 180 million voters. *Nature Human Behavior*, *5*, 998–1008. https://doi.org/10.1038/s41562-021-01066-z

Gavrila, D. M. (1994). R-tree index optimization. In T. Waugh & R. Healey (Eds.), *Proceedings of the Sixth International Symposium on Spatial Data Handling*, Edinburgh, Scotland (pp. 771–791). Taylor & Francis.

Goodchild, M. F. (2016). GIS in the era of big data. *Cybergeo: European Journal of Geography*, *20*, 27647. http://journals.openedition.org/cybergeo/27647

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *ACM SIGMOD Record*, *14*(2), 47–57. https://doi.org/10.1145/971697.602266

Han, J., Kamber, M., & Tung, A. (2001). Spatial clustering methods in data mining: A survey. In H. Miller & J. Han (Eds.), *Geographic data mining and knowledge discovery* (pp. 177–206). Taylor & Francis. https://doi.org/10.4324/9780203468029_chapter_8

Liu, J., Li, H., Gao, Y., Yu, H., & Jiang, D. (2014). A geohash-based index for spatial data management in distributed memory. In *Proceedings of the 22nd International Conference on Geoinformatics*, Kaohsiung, Taiwan (pp. 1–4). IEEE. https://doi.org/10.1109/GEOINFORMATICS.2014.6950819

OmniSci. (n.d.). *Omnisci overview*. https://www.omnisci.com/platform

Yue, P., & Jiang, L. (2014). BigGIS: How big data can shape next-generation GIS. In *Proceedings of the Third International Conference on Agro-Geoinformatics*, Beijing, China (pp. 1–6). IEEE. https://doi.org/10.1109/Agro-Geoinformatics.2014.6910649

Zhou, C., Lu, H., Xiang, Y., Wu, J., & Wang, F. (2020). GeohashTile: Vector geographic data display method based on geohash. *ISPRS International Journal of Geo-Information*, *9*(7), 418. https://doi.org/10.3390/ijgi9070418

How to cite this article: Kakkar, D., Lewis, B. & Guan, W. (2022). Interactive analysis of big geospatial data with high-performance computing: A case study of partisan segregation in the United States. *Transactions in GIS*, *26*, 1633–1641. https://doi.org/10.1111/tgis.12955