Article

Developing the Geospatial Big Data Benchmark: A Comparative Framework for Evaluating Raster Analysis on Big Data Platforms

David Haynes,[1*] Philip Mitchell,[2] and Eric Shook[3],

[1] Institute for Health Informatics, University of Minnesota; dahaynes@umn.edu
[2] Ali I. Al-Naimi Petroleum Engineering Research Center, King Abdullah University of Science and Technology, Kingdom of Saudi Arabia; mitc0415@umn.edu
[3] Geography Environment and Society; University of Minnesota, eshook@umn.edu
* Corresponding author

Abstract: Technologies around the world produce and interact with geospatial data instantaneously, from mobile web applications to satellite imagery that is collected and processed across the globe daily. Big raster data allows researchers to integrate and uncover new knowledge about geospatial patterns and processes. However, we are also at a critical moment, as we have an ever-growing number of big data platforms that are being co-opted to support spatial analysis. A gap in the literature is the lack of a robust framework to assess the capabilities of geospatial analysis on big data platforms. This research begins to address this issue by establishing a geospatial benchmark that employs freely accessible datasets to provide a comprehensive comparison across big data platforms. The benchmark is a critical for evaluating the performance of spatial operations on big data platforms. It provides a common framework to compare existing platforms as well as evaluate new platforms. The benchmark is applied to three big data platforms and reports computing times and performance bottlenecks so that GIScientists can make informed choices regarding the performance of each platform. Each platform is evaluated for five raster operations: pixel count, reclassification, raster add, focal averaging, and zonal statistics using three different datasets.

Keywords: geospatial; computation; spatial benchmark; cybergis

1. Introduction

We are in the era of big raster data. Planet, formerly Plant Labs, has a constellation of over 200 satellites that collect 1.4 million images each day.[1] Satellite imagery or earth observation data produces petabytes of data yearly and organizations like the Intergovernmental Panel on Climate Change (IPCC) produce simulated raster datasets that are multiple petabytes.[2]

The volume of data available for geospatial researchers is growing and, yet we do not have a standardized set of tools and best practices for accessing, manipulating, and performing analyses on big geospatial data. Currently, researchers develop novel ways for downloading data onto their workstations to perform their analyses. The reliance on this workflow reduces the ability to scale geospatial analyses on large datasets. However, it is necessary because geospatial data is different. Geospatial data is different because it has relationships and those relationships need to be considered when analyzing the data. Our research examines if big data platforms that support geospatial occur differential computational performance costs for preserving those relationships.

1.1. Big Geospatial Data

GIScience has co-opted new technology for geospatial problem-solving. In particular, the rise of distributed platforms has facilitated parallel spatial computation. Generally speaking, there are three types of big data architectures currently supporting big geospatial data: parallel databases, No-SQL databases, and the family of Hadoop based platforms which implement map-reduce either in memory (Spark) or to disk (Hadoop). Haynes[3] provides a complete description of platform architectures that support raster data including PostgreSQL, SciDB, RasDaMan, Hadoop-GIS, SparkArray, and GeoTrellis.

While there are a number of platforms that support raster data, there is little evidence describing capabilities and performance of spatial analysis operators on big data platforms. This is problematic as there is a need for the GIScience community to be able to use these platforms to process big raster data. The performance of raster operations has not been adequately addressed within the big data community. Instead, the literature surrounding big raster data platforms has three themes 1) novel implementations, 2) domain-science implementations, and 3) meta-reviews.

1.2. Novel Implementations

Novel implementations are characterized by the extension of a new platform to support geospatial data and methods. Examples of this are Palamuttam[4] and Wang[5] who developed a multi-dimensional array libraries for Apache Spark. Li and colleagues[6] developed, FASTDB, a distributed array database. Other research groups have extended SciDB to support spatial analysis.[7–9] These papers focus on specific problems, have small use cases, and are implemented on specific datasets. In comparison, with a benchmark their implementations are too specific and cannot be translated or adopted by the broader community.

1.3. Domain Science Implementations

Domain-science implementations focus on the development of a spatial workflow on a big data platform. This work fills a critical need in the literature as it demonstrates the general usability of the platform.[10–13] While these papers, are widely helpful for understanding the capability of a particular platform, they limited in their ability to compare workflows across platforms when compared with a benchmark.

1.4. Meta Reviews

Meta-reviews discuss the platforms and their capabilities.[14–18] These papers are orientated towards the broadest communities and focus on the platform capabilities not platform performance. They describe the directions and trends that are on the horizon and indicate how their adoption can fulfill larger goals within the broader community. Yet these too are limited in when compared to a benchmark as they offer no results to compare different systems.

1.5. Need for a Geospatial Benchmark

A gap within the geospatial high-performance computing community is the lack of a benchmark, which allows for the evaluation of spatial operators across platforms. Ray and colleagues[19] implemented Jackpine as a benchmarking framework for vector data. Jackpine implemented nine spatial analysis operations and five topological relationships on three vector data types: point, lines, and polygons. Rabl and colleagues[20] implemented benchmarking for array databases but only tested data loading and array subsetting operators. Therefore, a spatial benchmark that tests raster operators is needed. This benchmark will aid the geospatial community by providing publicly available reference datasets and methods that scholars could use for comparisons.

This research will establish a benchmarking framework that provides a means to accurately compare the performance of raster spatial analysis on big data. Our framework examines the performance of local, focal, and zonal operations on each platform. While it is impossible to test every potential operation, our framework examines five operators that have broad use cases. In addition, the benchmark evaluates system performance with both volume and variety characteristics.

The geospatial benchmark will provide insights into the complexity of performing big data spatial analysis. The benchmark is needed to 1) identify operator performance issues, 2) determine if the underlying causes are related to the architecture or implementation, 3) identify areas where research is needed.

2. Geospatial Big Data Benchmark

Benchmarks are defined as a dataset, a platform(s), and a series of operations. The definition of a geospatial benchmark is similar. First, we define dataset by a consistent spatial extent. As there are no previous existing datasets that could be employed as the benchmark, we employ multiple datasets with increasing spatial resolutions to determine performance of each platform. We employ three different platforms in this analysis, and employ three of the major spatial operations used in raster analyses.

2.1. Data

We use three different datasets in our analysis (Table 1). Each of these publicly available datasets has been clipped to the spatial extent of the continental United States. Each dataset used in this analysis is classified satellite imagery. They have been chosen due to their widespread availability and use within geospatial applications. We have chosen three different datasets that represent increased levels of spatial granularity. Table 1 reports the number of pixels that are present in each dataset. Table 1 should be used by geospatial researchers as a guide to compare their project and determine the performance levels they should expect based upon the platform that they have chosen.

Table 1. Raster Dataset Description.

Dataset Name	Spatial Resolution	Pixel Size in Meters	Total Pixels
GLC	.0089 decimal degrees	1KM^2	18 Million
MERIS	.0027 decimal degrees	300M^2	186 Million
NLCD	30 meters	30M^2	1.69 Billion

To benchmark zonal operations, we also include three vector datasets of the continental United States. We used state, country, and census tract cartographic boundaries from the US (Table 2). All datasets have the same spatial extent, and the only difference is the number of features present in the dataset. As neither GeoTrellis or SciDB read shapefiles, the datasets were converted into Geographic JavaScript Object Notation (GeoJSON).

Table 2. Vector Dataset Description.

Dataset Name	Shapefile (mb)	JSON (mb)	Number of Features
States	3 megabytes	5 megabytes	49
Counties	15 megabytes	24 megabytes	3,108
Census Tracts	700 megabytes	1.7 gigabytes	64,882

2.2. Spatial Partitioning

As with all big data platforms partitioning the data is an important and necessary step. Data partitioning, in particular, is critical to big data platforms as it is one way to tune the platform to hardware and the data. In this research, we tuned the platforms by using a defined set of data partition sizes. The term tile size is used to designate the raster partitioning scheme. A tile size of 50 means that there are 2,500 pixels (50x50) within a single partition of the data. Tile sizes have been standardized across platforms for maximum comparability of performance. Each platform is evaluated across a series of tile sizes that will be optimal or sub-optimal. Not all platforms are compared with all tile sizes, and the decision to limit these is due to the minutes changes in performance on the platforms based on tile sizes. For example, SciDB shows minor performance differences unless the tile size varies by 1 million pixels.

2.3. Spatial Operations

Our analysis focuses on three classes of raster operations: local, focal, and zonal. Table 3 provides a simple matrix of how each of the platforms and how spatial operators are translated and performed on the given dataset. There are two types of evaluation: lazy and eager. Eager evaluation, which has historically been more common, is an operator that once called performs the analysis on the dataset. The age of big data platforms has resulted in the rise of lazy evaluation, which only operates on the data when the result of that operation is needed. However, this makes a comparison between platforms with different evaluations complex. Therefore, when a platform lazy evaluates, we include an additional step that forces the evaluation to complete. The eager method is used to ensure consistency between data platforms.

Table 3. Description of Raster Operations by Platform with Evaluation Type

Function	Class	Description	PostgreSQL	GeoTrellis	SciDB
Count Pixels	Local	Counts all pixels in the raster of a given value	Eager	Eager	Eager
Reclassify	Local	Changes all occurrences of a value in a raster to a new value	Eager	Lazy	Lazy
Raster Add	Local	Adds two rasters together	Eager	Lazy	Lazy
Focal Mean	Focal	Calculates the focal mean of a 3x3 neighborhood	Eager	Lazy	Lazy
Polygonal Summary	Zonal	Calculates statistics for each polygon or multipolygon of a vector layer overlaid on a raster layer	Eager	Lazy	Eager

2.4. Local Operations

The class of local raster operations operates on each cell individually without reference to the surrounding cells. This type of operation lends itself to parallelization because the dataset once partitioned can be operated on independently. In our benchmarking framework, we use: pixel count, reclassification, and raster add. While local operations act on a cell individually, they operate on the entire dataset.

2.5. Pixel Count

The pixel count operation returns the number of occurrences of a given pixel value within a raster. This function is the basis for any histogram like function, in which the dataset must be traversed and information gathered regarding the cell values that have been defined. Identification of pixels by value is of particular importance for land cover change analyses. Additionally, we utilize this method because it allows us a standardized method for forcing lazily evaluated operations to become eager across platforms.

In this application, each pixel value represents a land cover type, and the operation will return the number of times this value occurs. As the function must traverse the entire dataset, there is no performance gain or loss when a pixel value is frequent or rare within the dataset.

2.6. Reclassifcation

Reclassification identifies pixels of a given value and changes them to a new value specified by the user. Reclassification is a specific case of map algebra, in which a pixel value is evaluated and then replaced with a new value. There are two possibilities for reclassification operators in PostGIS 1) map algebra operator and 2) reclassification operator. We empirical compared the ST_Reclassify and ST_MapAlgebra operators and determined that ST_Reclassify is the faster operator. Both GeoTrellis and SciDB are agnostic about the difference between reclassification and map algebra. Our implementation of reclassification on GeoTrellis utilizes the "local if" function. SciDB's implementation of reclassification is an "if-then-else" statement. The pixel count operator was called post-reclassification for SciDB and GeoTrellis.

2.7. Raster Add

Raster add is a map algebra operation. It takes two rasters as inputs and adds the values of each cell and returning a new raster. In our study, datasets were single band rasters, and we used the same dataset as the first and second raster. Raster add is lazily evaluated in GeoTrellis and SciDB, so the count pixel function was called to force evaluation.

The raster add function is used for methods such as the Normalized Difference Vegetation Index (NDVI). Calculation of NDVI requires two different bands, near-infrared and red, which go through a local mathematical process to return the actual value. The raster add function tests the ability for each platform to join these large datasets and return a value.

2.8. Focal Operations

Focal functions differ from local functions in that the output values are influenced by surrounding cells. A kernel or window is used to specify the size of the analysis or how many adjacent pixels are needed to determine the output value. A focal analysis is a vital tool when performing geospatial analyses. They are predominantly used in computations that involve smoothing or interpolation. For example, removing vegetation pixels from a bare earth Digital Elevation Model.[21] Additionally, focal operators are complex because if the dataset is distributed the operator must employ a systematic approach for locating adjacent tiles and pixels.

2.9. Zonal Operations

Zonal functions, specifically polygonal summaries, are a complex analysis as they involve both raster and vector datasets. Spatially irregular zones (i.e., Hawaiian islands) lead to increased complexity. Therefore, when a calculation is applied to a specific zone only a subset of the dataset must be processed. Ding and Desham[22] define this problem as loosely-synchronous.
For the benchmarking framework, we tested the concept of polygonal summaries of raster datasets. Each vector dataset contained both polygons and multipolygons at decreasing scales (e.g., states, counties, and tracts). The operators calculated the minimum, maximum, and average value within each zone. Polygonal summary statistics are applied in a wide variety of analyses such as phytoplankton blooms and determining the effect of vegetation cover on soil loss.[23,24] In many cases, polygonal summaries are used to focus our knowledge on the spatial process and report relevant information to decision-makers.

3. Big Data Platform Comparison

3.1. Platform Descriptions

Much of the literature that develops improvements in big data platforms use customization, such as the development of additional methods or tuning of the software to support specific hardware. Our approach differs in that we focus on the development of a geospatial benchmark for spatial analysis. Then we apply the benchmark to platforms and evaluate their ability to perform spatial analyses.

3.1.1. PostGIS 2.4 (PostgreSQL 9.6)

PostgreSQL is a relational database that has supported spatial data types since its initial release of PostGIS in 2001. The raster datatype was added as an official datatype of PostGIS in 2012. A raster dataset, once loaded into PostgreSQL, assumes a table representation consisting of two columns: RID and Raster. The RID column is a primary key and the raster column contains the binary pixel value data, which is stored in Binary Large Object (BLOB) and can only be accessed by using PostGIS raster functions.

3.1.2. SciDB 16.9

SciDB is an open-source multi-dimensional array database designed by Dr. Michael Stonebraker.[10,25] Its development was spurred by the concept that many scientific datasets have array-like structures, and there are costs to restructuring the datasets to persist as arrays within a relational database. SciDB's platform uses arrays as the primary data structure and has been co-opted by the geospatial community.[9,17,26,27] SciDB's massively parallel processing (shared-nothing parallel database) architecture allows it to process multi-dimensional arrays or geospatial imagery that are multiple petabytes. [10,25]

SciDB is not the only array database platform. RasDaMan, developed by Dr. Peter Bauman, is specifically designed to work with raster datasets.[28] We have chosen SciDB because SciDB's community edition can be extended to multiple instances or nodes, whereas only the enterprise version of RasDaMan supports this.

Loading large raster datasets into SciDB can be a monumental task, as the primary data structure of SciDB is a one-dimensional array with one or more attributes. However, satellite imagery is two dimensional with at least one attribute. Currently, SciDB does not have built-in capabilities for reading in geo-referenced imagery. Community efforts to add geospatial functionality into SciDB have occurred, but nothing has been formally.[7,29]

3.1.3. GeoTrellis 1.2 (Apache Spark 2.1)

Apache Spark is an open-source high-performance distributed computing environment that began in 2009 in the UC Berkeley RDAD lab. Apache Spark is a component of the Hadoop ecosystem and has been shown in some operations to be 20x faster than Hadoop.[30] This improved performance is due to Apache Spark holding data in memory and conducting operations in memory instead of writing to disk as Hadoop does.

There is a growing literature examining multi-dimensional arrays on Apache Spark. Wang and colleagues[5] implemented a new array datatype, SparkArray, to load and process raster data. Doan and colleagues[15] compared the performance of loading and subsetting operations between SciDB and Apache Spark. The GeoTrellis library is one of the first libraries to go beyond simple array datatypes, as it has been developed for processing, visualization, and analysis of geospatial data.

GeoTrellis began as a research project of Azavea in 2006, however in 2013, the GeoTrellis project became a member of the Eclipse Foundation, and was redeveloped in Scale and used Apache Spark as its distribution and processing engine. The GeoTrellis library implements Paired Resilient Distributed Datasets (RDD) for spatial datasets, in which each Paired RDD is represented as a key and value pair. The key refers to a specific geographic location of the raster data a tile and the value is a multi-dimensional matrix.

3.2. Hardware

We use the Extreme Science and Engineering Discovery Environment (XSEDE) cyberinfrastructure to provide computation resources for our computation environment.[31] XSEDE is a National Science Foundation investment that allows for requests of high-performance computation allocations. Supercomputer Wrangler fulfilled our request at the University of Indiana, which allocated three compute nodes and installed our software (TG-SES160012).

Each node was a Dell PowerEdge R630, equipped with two Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz processors, with 12 cores for each Xeon. Additionally, each node has 128 Gigabytes of 2133 MHz DDR4 RAM. While each node had 24 cores, all performance times utilized 12 cores for SciDB and Apache Spark. PostgreSQL at this time has a one to one relationship between queries and cores. All platforms utilized a Lustre backend for large data storage. The Lustre version was Lustre 2.5.5, and the Lustre client was 2.10.3. There was a small discrepancy in operating systems, as SciDB 16.9 at the time of this was not properly configured to run CentOS 7. Therefore, SciDB was built on CentOS 6 OS, while both Apache Spark and PostgreSQL were configured to run in a CentOS7 environment.

4. Results

The results represent a selection of the analyses we performed for demonstrating our benchmarking framework. The results depict broad characterizations that users should expect when performing spatial operations on the platforms.

In order to effectively assess the effect of tuning into the results, each operator is performed on each dataset with various tile sizes. By varying the tile size at regularly defined intervals, we can determine when the optimal performance of a particular platform occurs on a particular dataset. The full range of tile sizes was not be applied across each platform due marginal changes in performance. However, we have identified a selection of tile sizes that allows for a fair comparison across platforms and characterize optimal performances of raster analyses on big data platforms.

We report the averaged time, which determined from three consecutive tests. Speed-up is determined by taking the best performing time for PostGIS and comparing it with the best performance time for SciDB and GeoTrellis. The best performance time is used as this represents the optimally tuned performance for each platform for each dataset. Since all queries for PostGIS are single-core, speed-up per-core is determined by dividing speed-up by the number of processors or instances used. For all analyses, this value was 12.his section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation as well as the experimental conclusions that can be drawn.

4.1. Local Operations

4.1.1. Pixel Count

Figure 1 reports a common issue when analyzing raster data on various platforms without a sensitivity test. Raster tile tuning is specific to the platform. Figure 1 shows that the tile size that works well on one platform should not be applied to another platform as it will result is likely to have suboptimal performance. For example, in Figure 1, a tile size of 1000 (1,000,000 pixels per tile) depicts some of the worst performance times for PostGIS and GeoTrellis, whereas for SciDB the performance is quite good. As tile sizes decrease, both PostGIS and GeoTrellis performances improve, whereas SciDB's performance degrades.

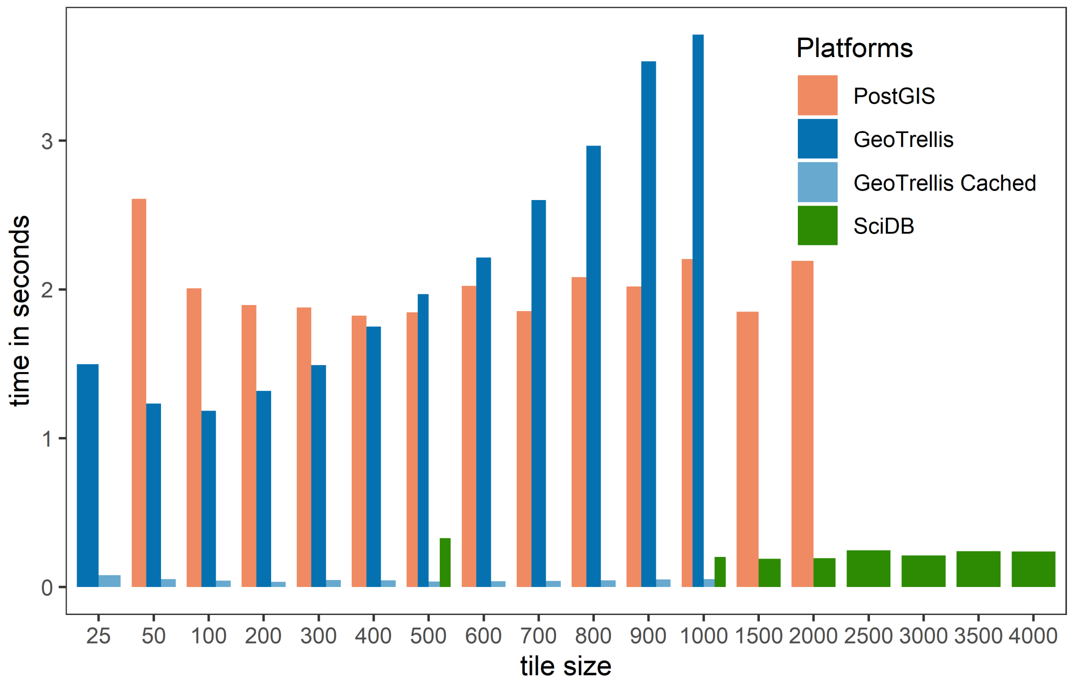

Figure 1. Performance of Pixel Count on dataset GLC on all Platforms.

The results of the Pixel Count operator are further shown in Table 4. Table 4 depicts the results from the smallest and largest datasets (i.e., GLC and NLCD), full results reported (Appendix Table 1). The results in Table 4 indicate that the dataset GLC (18 million pixels) is not big data. For big data platforms like GeoTrellis or SciDB, this is a small dataset, and there is minimal performance. However, as the volume of data increases there are notable performance gains with big data platforms (Table 4). Table 5, reports that speed-up per-core (2.49) is first achieved by SciDB on the Meris dataset (186 million pixels). SciDB's per-core improvement increases up to 5.7 times per core. The best GeoTrellis performances occur with the Cached reads, in which the data has been read once and is cached in memory.

Table 4. Pixel Count Performance Times in Seconds on all Platforms on Raster Datasets: GLC, NLCD

	GLC				NLCD			
Partition size	**PostGIS**	**GeoTrellis**	**GeoTrellis Cached**	**SciDB**	**PostGIS**	**GeoTrellis**	**GeoTrellis Cached**	**SciDB**
25		1.50	0.08			97.05	10.92	
50	2.61	1.23	0.05		1184.82	84.96	6.11	
100	2.01	1.18	0.04		1039.69	86.42	5.10	
200	1.89	1.32	0.03		1000.76	100.16	5.30	
300	1.88	1.49	0.05		994.96	119.04	5.77	
400	1.82	1.75	0.05		990.35	137.89	6.28	
500	1.84	1.97	0.04	0.33	991.59	167.46	6.94	17.60
600	2.03	2.22	0.04		996.43	196.55	7.45	
700	1.85	2.60	0.04		991.93	231.50	7.96	
800	2.08	2.97	0.04		999.22	258.02	8.46	
900	2.02	3.53	0.05		1005.01	309.73	9.12	
1000	2.21	3.71	0.05	0.20	1005.13	355.63	9.85	14.36
1500	1.85			0.19	1002.52			21.94
2000	2.44			0.19	1032.11			19.50
2500				0.24				19.86

3000	0.21	20.79
3500	0.24	21.87
4000	0.24	20.96

Table 5. Pixel Count Speed-up on all Platforms all Raster Datasets

Dataset		PostGIS	GeoTrellis	GeoTrellis Cached	SciDB
GLC	Best Time	1.823	1.183	0.034	0.189
	Speed-up Per-core		0.128	4.425	0.804
Meris	Best Time	12.526	10.010	0.094	0.428
	Speed-up Per-core		0.104	11.065	2.439
NLCD	Best Time	990.351	84.956	5.101	14.363
	Speed-up Per-core		0.971	16.178	5.746

4.1.2. Reclassification

The results of the reclassification operator, differ sharply from the pixel count operator. PostGIS's reclassification function works very efficiently and allows it to outperform the big data platforms (Figure 2). PostGIS reports faster performance times than GeoTrellis and SciDB on GLC dataset. Table 6 shows that only minimal speed-up per-core performance gains occur when using big data platforms. When examining the two smaller datasets (GLC and MERIS), GeoTrellis' compute time is slower than PostGIS, meaning the platform is penalized for using small data (Table 6). Speed-up occurs big data platforms with the largest dataset NLCD, but no per-core speed-up improvements overall.

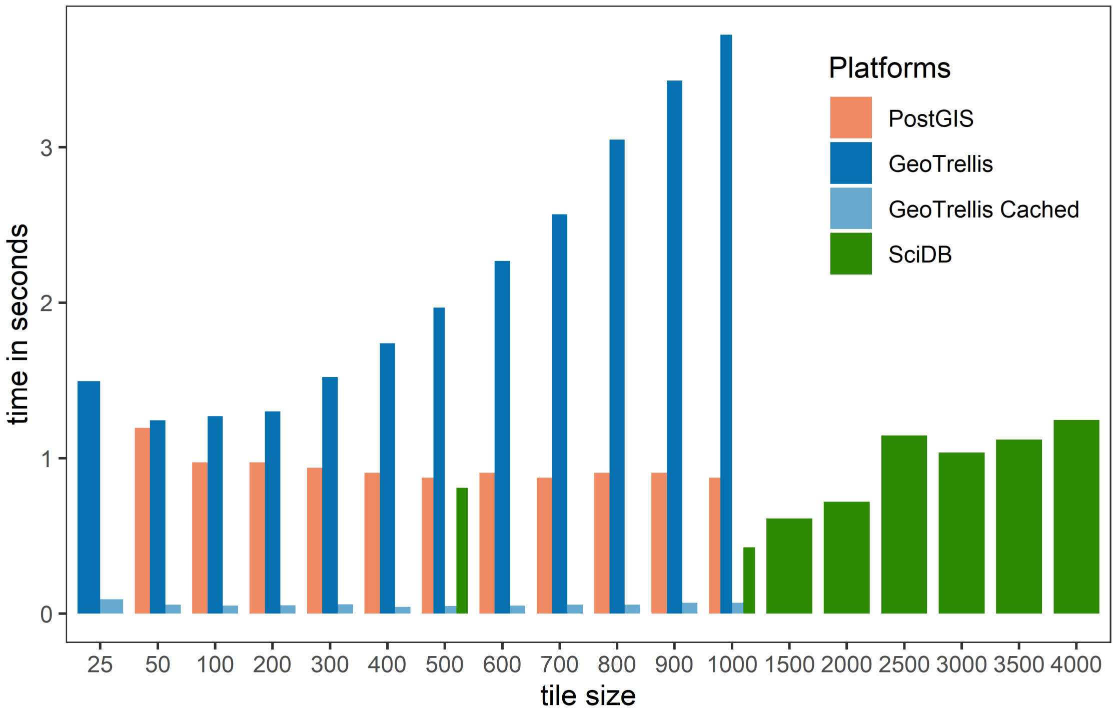

Figure 2. Performance of Reclassification on Dataset GLC on all Platforms

Table 6. Reclassification Speed-Up on all Raster Datasets on all Platforms

Dataset		PostGIS	GeoTrellis	GeoTrellis Cached	SciDB
GLC	Best Time	0.873	1.243	0.043	0.426
	Speed-up Per-core		0.058	1.691	0.171
Meris	Best Time	6.183	10.117	0.140	1.934

	Speed-up Per-core		0.051	3.689	0.266
NLCD	Best Time	532.619	95.230	29.156	137.296
	Speed-up Per-core		0.466	1.522	0.323

4.1.3. Raster Add

Table 7 describes the performance of the raster add function across all three platforms. The results of the raster add function are similar to the results in pixel count. The big data platforms have a tremendous performance advantage over PostGIS. Table 7 depicts that PostgreSQL's best performance occurs when the tile sizes are largest. As tile size increases performance increases, yet PostGIS was unable to ever successfully join the NLCD dataset at any tile size. The operation ran for over 24 hours without ever finishing. Both SciDB and GeoTrellis were able to join all raster datasets, and we found variation in join performance between the platforms.

Table 7. Raster Add Performance Times in Seconds on all Raster Datasets and all Platforms

	GLC				**NLCD**			
Partition size	PostGIS	GeoTrellis	GeoTrellis Cached	SCIDB	PostGIS	GeoTrellis	GeoTrellis Cached	SCIDB
25		1.62	0.14			209.74	38.79	
50	1160.67	1.15	0.09		*	171.04	19.03	
100	170.67	1.23	0.07		*	174.11	14.42	
200	56.87	1.34	0.08		*	201.12	14.43	
300	42.04	1.50	0.05		*	237.16	15.36	
400	35.43	1.74	0.04		*	273.04	15.87	
500	34.30	1.98	0.05	0.87	*	330.58	16.97	359.37
600	31.61	2.30	0.05		*	388.62	18.01	
700	31.76	2.73	0.06		*	457.46	19.37	
800	30.45	3.22	0.06		*	510.85	20.00	
900	30.63	3.52	0.07		*	614.18	21.07	
1000	30.28	3.93	0.08	0.83	*	701.00	22.26	370.79
1500				1.11				372.05
2000				1.60				377.71
2500				2.27				398.06
3000				2.62				395.28
3500				2.77				383.84
4000				3.25				400.82

* Unable to complete analysis

4.2. Focal Analyses

PostGIS performed well on the small and medium datasets (GLC and MERIS). However, it was unable to finish computations on the NLCD dataset (1.69 billion) pixels for any tile size (Table 8). For focal operations, the best performance occurred when the tile size is smallest at 50, as the tile size grew so did the time to complete the query.

Table 8. Results of Focal Analysis Performance Times in Second on GLC for all Platforms

Partition size	**PostgreSQL**	**Geotrellis**	**GeoTrellis Cached**	**SCIDB**	**SciDB overlap**
5		1.563	0.135		
50	118.883	1.272	0.124		
100	123.314	1.309	0.114		
200	126.583	1.393	0.109		
300	128.053	1.581	0.138		

400	127.472	1.821	0.146		
500	128.040	2.086	0.156	8.380	4.420
600	128.084	2.355	0.179		
700	129.083	2.689	0.188		
800	128.354	3.263	0.208		
900	128.390	3.524	0.272		
1000	128.422	3.828	0.275	14.148	8.243
1500				19.708	7.223
2000					

SciDB's performance on focal operations was challenging, and we provide full performance values (Appendix Table 3). The irregular performance times occur on large dense datasets with partition sizes greater than 1500x1500 pixels. Performance times of the overlapped array using the focal operator are two times faster than the standard array. This decrease in performance is due to SciDB's query planner detecting that the array is not structured for parallel implementation and initiating a redimensioning step, which is computationally expensive.

GeoTrellis' performance for focal operations is superior to SciDB and PostgreSQL. The performance improvements are evident with small and large datasets. For example, GeoTrellis achieves a 7x speed-up per-core on the smallest dataset GLC. GeoTrellis' performance continued to improve as it reached 9x speed-up per-core on Meris and finished NLCD. While the performance gains of GeoTrellis over SciDB are not as large, the ease and overall performance make GeoTrellis the superior platform.

4.2. Zonal Analyses

Unlike the other operations, PostGIS provides the best overall performance on zonal operations across datasets (Table 9). The full results of zonal operations are reported (Appendix Table 4). PostGIS's superior performance relies on built-in operations that support both vector and raster data types. PostGIS' rasterization process is serial but performs very efficiently. Additionally, the function employed for conducting zonal statistics with PostGIS, ST_SummaryStatsAgg, is an aggregate function. Meaning it operates on each geographic feature and raster tile independently. A major item of concern is the "U" shape performance curve that PostGIS creates (Haynes et al. 2017). We find that these trends are consistent across all datasets. The optimal performance tile size varies as the number of features increases and the geographic extent of the features decreases.

Table 9. Zonal Operator Performance Times in Seconds on GLC for all Platforms with State Boundaries

Partition size	PostGIS	GeoTrellis	SCIDB - All Operations	SCIDB - Join Only
25		257.761		
50	5.688	69.212		
100	3.886	21.169		
200	3.846	7.773		
300	4.326	5.489		
400	4.698	4.601		
500	4.825	4.314	16.800	1.375
600	4.993	4.330		
700	4.999	4.391		
800	5.070	4.573		
900	5.187	4.811		
1000	5.429	4.964	16.808	1.597
1500			16.152	1.951

2000	16.659	2.366
2500	18.404	3.397
3000	23.526	4.289
3500	24.544	5.033
4000	26.491	5.550

The performance results for SciDB are mixed. Figure 4 shows the evidence that SciDB is potentially a good platform for performing zonal statistics. Figure 4A depicts the performance between SciDB and PostgreSQL on the states dataset, in which the PostgreSQL easily outperforms SciDB-All Operations. However, Figure 4B shows that as the number of features increases SciDB becomes the superior platform. Unlike PostGIS or GeoTrellis, with SciDB there is relatively little change in performance as chunk size increases.

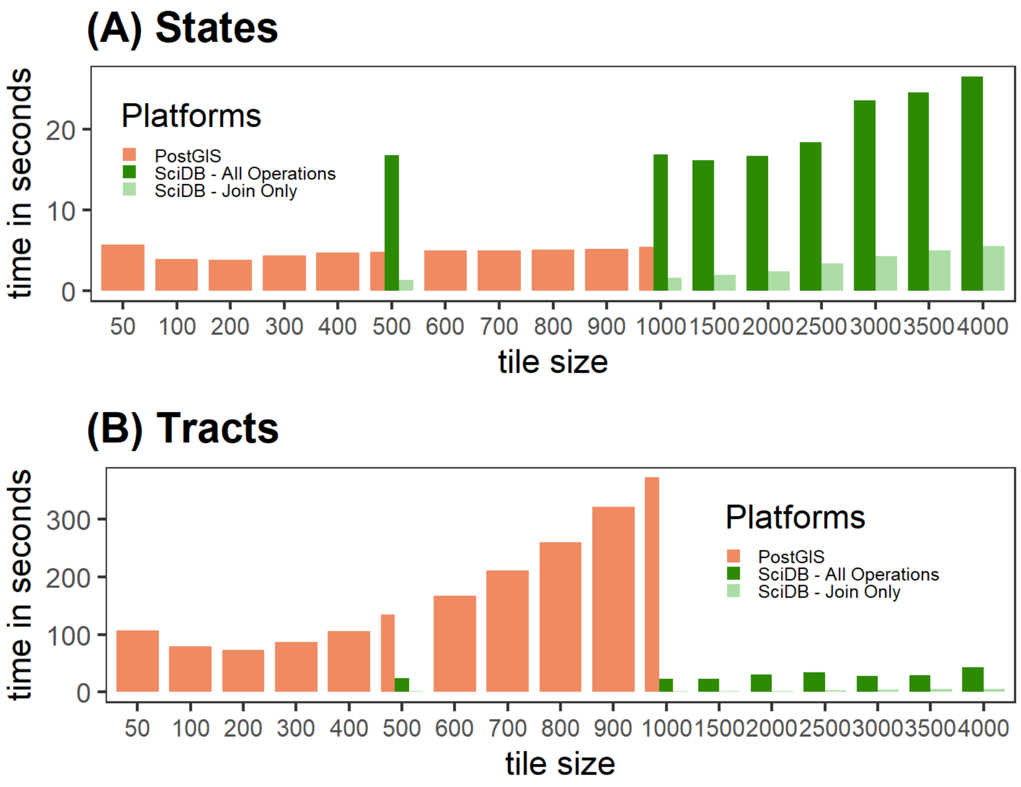

Figure 4. SciDB and PostGIS Zonal Operator Performance Time on GLC with States and Tracts.

Figure 4 reports results for SciDB that are both "Join-Only" and "All-Operations." The join-only, performance times assume that the masking process has already been performed. This is an unlikely assumption and full time results are in Appendix Table 5.

The results of GeoTrellis are surprising (Table 10). GeoTrellis 1.2 does not have a feature that would allow it to take a collection of geometries and perform the operation across all features. Therefore, the operation is serial, which results in poor performance. Table 10 depicts that better performance occurs with larger tiles.

Table 10. GeoTrellis Zonal Operator Performance Time in Seconds on GLC for States and Tracts

Partition size	States	Tracts
25	257.761	40685.147
50	69.212	12982.878
100	21.169	4897.609
200	7.773	2205.593

300	5.489	1747.577
400	4.601	1579.983
500	4.314	1486.745
600	4.330	1571.242
700	4.391	1647.180
800	4.573	1726.475
900	4.811	1881.314
1000	4.964	1909.860

4. Discussion

4.1. Local Operators

Overall, big data platforms are superior when conducting local operations. The big data platforms specialize in partitioning large datasets and analyzing them in parallel. The data structures and architectures employed by both platforms are likely to perform well on geospatial computational work. SciDB's array-store architecture in many cases outperformed GeoTrellis and PostGIS at fetching data quickly, which is a primary benefit of the platform. The results of the pixel count operation depict the computational advantages of using a big data platform. However, the results of the reclassification operation demonstrate the benefit of having optimized functions instead of generalized functions. The big data platforms use generic if then else operators, whose performance are slower on small datasets and did not have much performance gain on larger datasets.

The results of the benchmark for the raster add operation were striking. In comparison to all the other local operations, the raster add operation depicts the most substantial performance gains when using a big data platform like SciDB or GeoTrellis. PostGIS cannot process the join between these two large raster datasets (i.e., NLCD). Table 7 also contains interesting results when comparing the performance of between SciDB and GeoTrellis. Initially, SciDB performs the best on the GLC and Meris datasets with 18 million and 186 Million pixels respectively. SciDB's best performances are about twice as fast as GeoTrellis for each dataset. However, GeoTrellis' performance is superior on the NLCD dataset with 1.69 billion pixels. GeoTrellis' best performance of 171 seconds being twice as fast as SciDB's at 359 seconds.

This change was unexpected. The query that is implemented by SciDB is a series of lazy operations, whereas GeoTrellis submits a single lazy operation. For SciDB the order of operations is the following:

1. first a lazily evaluated join operation,
2. followed by a lazily evaluated apply operation,
3. followed by a lazily evaluated filter operation,
4. and ultimately concludes with the eager aggregation.

In GeoTrellis, map algebra operations are reduced to r1 + r2 = outdataset. This simple equation combines all of these steps and returns a lazy new RDD. To get the results of this RDD, we must force execution we utilizing the count pixel function we have previously tested.

The reason we observe the small changes in the performance of these platforms lies in the architecture of the platforms. The observed differences are related to an imperfect ratio between SciDB instances and the number of data partitions available for each instance of SciDB, which is a classic load balancing problem. For example, if we have 12 SciDB instances and 24 chunks and the operation takes 10 seconds to complete for a chunk. It will take SciDB 20 seconds to finish this operation with 24 chunks and 12 instances. If we have the same dataset partitioned into 26 chunks, it will take SciDB 30 seconds.

The approach used for the Apache Spark framework is different because the platform is agnostic to the ratio of cores to the number of data partitions because the data is all in memory. The Apache Spark application defines the number of cores available and assigns data to available cores. Our results show that while SciDB suffers from an imperfect load balance on all datasets, its ability to fetch the data is outstanding and allows it to outperform GeoTrellis on smaller datasets. However, as the dataset and load imbalance grows, Apache Spark's framework performs better because it can avoid load balance issues.

4.2. Focal Operators

Creating comparable methods for the focal analyses was complex, but our results indicate that there are apparent differences in the platforms.

Focal operations should be avoided when using PostGIS as it provides unsatisfactory results; due to the resulting dataset from PostGIS not being equivalent to the resulting dataset from SciDB or GeoTrellis. The PostGIS focal operator is not an aggregate. Instead, it operates on each tile independently. The focal operation is used in concert with PostGIS's MapAlgebra function. Therefore, pixels that are located on the edge of a tile will not have their focal value determined by cells adjacent to it in the next tile. This has potentially serious implications for large datasets like NLCD when it has many small tiles. The alternative is to use the ST_Union operator to merge all of the tiles and then perform the focal operator. Merging tiles is unlikely to be unsuccessful as a raster dataset size can exceed the PostgreSQL row memory limit. Haynes[17] indicates that the ST_Union operator is computationally intensive and degrades query performance.

SciDB also presented some unexpected challenges. Overlapped arrays are not a unique class of arrays; they are an additional specification of the array schema. A SciDB array can be defined with an overlap in any dimension.[25] The overlap allows SciDB each array to have data from adjacent tiles. An array with a defined overlap value will now be able to conduct a focal operation in parallel.

Wide-ranging performances were encountered when using the focal operator, highlighting architectural issues using large partition sizes (i.e., greater than 1500x1500 pixels) on big arrays. The window operator, for SciDB, works on a tile sequentially and stores them in memory. This process exhausted the memory on 128 Gigabyte the node. Queries that used tiles sizes of 2000 or greater caused the query to hang, even if completed, and sometimes resulted in the SciDB needing to be rebooted.

While both big data platforms are better alternatives to PostGIS, GeoTrellis is the best performer. When a focal operation is initiated on the RDD, GeoTrellis is aware of the spatial arrangement of every tile of the dataset. It then collects all edge pixels that are necessary for each tile and shuffles them to the neighboring tiles, making the focal operation embarrassingly parallel.

Another implementation advantage of GeoTrellis over SciDB is that it currently caches at the tile level. The first time the focal operation is initiated on a tile, GeoTrellis knows very little information about its adjacent pixels. Therefore, it must gather information about the adjacent pixels. When the operator moves to an adjacent pixel, it already has information about ⅓ of the adjacent cells from the previous query. GeoTrellis' implementation allows it, through caching, to reuse data that it has already read, speeding up the operation. Currently, on SciDB, the focal operator does not cache. Researchers have written additional operators for SciDB that improved the performance of the focal operation.[8] Unfortunately, the operator is currently depreciated and could not be tested within our framework. The concept of within tile caching should be applied to any focal operation on a big raster data platform.

4.3. Zonal Operators

Creating comparable methods for the zonal analyses was the most complex as it involved two different datasets vector and raster. Both PostGIS and GeoTrellis natively support vector data types. However, SciDB does not, which is problematic for performing zonal operations.

Zonal operators are an area in which both big data platforms struggled in comparison to PostGIS. Polygonal summaries are an area in which research is needed for big data platforms. One potential reason is that polygonal summary or zonal statistics tend to be viewed as a single operation, when in fact there are a series of steps.

1. rasterize the vector dataset to the same geographic extent as the raster dataset,
2. spatially join the masked raster and original raster dataset,

3. conduct an aggregation (i.e., min, max, sum, mean) between the two joined datasets.

Within each platform, polygonal summarization works differently. For PostGIS polygonal summarization process is serial. It takes each polygon and intersects it with spatially aligned tiles. It then rasterizes and joins the two datasets together and returns the requested statistics.

GeoTrellis avoids rasterization and uses a scan-line algorithm to identify the pixels of interest it then creates and returns a new RDD. The primary issue for SciDB is the development of an external process for creating and loading a masked dataset. Rasterization is a slow process that creates a second dataset that must be loaded into the platform, decreasing the performance substantially. Why perform the rasterization process and create a second potentially large dataset? We would argue that the implementation of a scanline operation would be beneficial for SciDB. Parallel scanline operations have been widely implemented within the literature.[32,33]

Solving the polygonal summary issue for GeoTrellis (Apache Spark) is complex. Our implementation utilized a loop which is used to iterate through all the features of the vector dataset. A function that performed this analysis for all features within the vector dataset is likely to improve the performance greatly.

However, there may be underlying architectural issues that could potentially reduce the overall performance of this operation. Zonal Statistics utilizes heterogeneous datasets, which causes issues for Apache Spark. Its typical way of handling this is to shuffle the data. However, the amount of shuffling implemented during zonal analysis is of great concern. Shuffling is necessary because Apache Spark does not join or merge heterogeneous datasets. For example, take a raster dataset with lots of tiles and scatter them across a series of nodes and then do the same thing with a vector dataset with lots of features. The resulting analysis incurs greater computational penalties when shuffling the data. The results in Table 10 support this. The performance increases when using a dataset with few features like states compared to a dataset like many features such as census tracts. GeoTrellis spends much of its time shuffling data around so that it can match the vector and raster datasets together. GeoTrellis' performance when there are fewer raster tiles and fewer vector features.

Yang[34] discusses a potential solution called Map-Reduce-Merge, which is designed to handle heterogeneous datasets. The implementation of such a feature for geospatial analysis is not without challenges as Apache Spark works best with small partitions. Currently, GeoTrellis partitions at the polygon level; this is problematic because polygons can span multiple tiles. To make the data more homogenous, vector-specific partition needs to be implemented. Partitioning will need to be implemented at two levels. The first level of partitioning is at the polygon and the second level of partitioning would be such that it breaks polygons into fragments that align with the raster partitioning structure. Afrati and Ullman[35] build upon this work discussing a similar strategy, map-reduce-join. The map-reduce-join strategy allows for the joining of heterogeneous datasets based upon star join. In which the smaller dataset (vector) is replicated to all potential matching tiles, and then joins are performed. GeoSpark implements a similar partitioning strategy when joining vector datasets in Apache Spark.[36]

5. Conclusions

This research develops the first Geospatial Big Data Benchmark that can be used for comprehensively comparing raster analysis on big data platforms. It provides a broad overview of a selected group of big data platforms and applies them to three classes of spatial operators. The development of the geospatial benchmark for raster operations is necessary and will aid the development of big raster data platforms. The benchmark guidelines are a critical piece component of the spatial infrastructure and the big spatial community and will provide the geospatial community with a reference tool for evaluating big raster platforms. The utility of the benchmark is demonstrated in the application of the benchmark to three existing big data platforms and their potential application to processing big geospatial data.

Table 11. Evaluation of Platform Performance on Raster Operations.

Local	Focal	Zonal

PostGIS	Moderate Performer	Poor Performer	**Top Performer**
SciDB	**Top Performer**	Poor Performer	Moderate Performer
GeoTrellis	**Top Performer**	**Top Performer**	Poor Performer

Table 11 reports an overall assessment of platform performance on big spatial operations. Both of the big data platforms SciDB and GeoTrellis exhibited superior performance methods on the local operations. In particular, map algebra operations are an area where these platforms demonstrate their superior performance, as PostGIS was unable to finish the computation for the raster add operator as the data volume grew. GeoTrellis was also the superior platform when performing focal operations. GeoTrellis has implemented various levels of caching that results in good performances on datasets of all sizes. Additionally, both big data platforms have the ability to restructure the data into an embarrassingly parallel format on demand. Lastly, big data platforms need of additional development work to support zonal operations. Both SciDB and GeoTrellis produced subpar performances for zonal operations. Of the two platforms, SciDB's performances were the same or similar to PostGIS on the medium and small datasets. SciDB's major bottleneck is the need to rasterize and load the external dataset, which is problematic for large datasets. While GeoTrellis has implemented methods to avoid rasterization, the current architecture's inability to match heterogeneous datasets without lots of shuffling limits the platform's use on zonal operations that contain a large number of vector features or have a small tile size. Table 11 highlights a second significant issue; none of the platforms we analyzed were successful at all of the three classes of raster operations. GeoTrellis was the top performer in two of the three categories, making it an ideal platform for developing spatial workflows.

5.1. Limitations

While this study has attempted to be very thorough and robust in its analysis, there are limitations to this research. The first limitation is that we have not examined every big data platform that currently analyses raster data. We have examined a selection of platforms that are documented in the literature and utilize different big data architectures: relational databases, No-SQL array databases, and Apache Spark (in-memory Hadoop). A second and related limitation is that new versions of the software will change the performance of these platforms. This is true, however again we reference the architectural limitations that we have identified. Unless there are major changes within the architecture, many of these problems will still exist. The last limitation is that we have not extensively examined the V's of big data. While this is also true, the benchmark provides the foundation on which a more extensive benchmark should be built.

5.2. Future Work

This research addresses a significant gap in the literature through the development of a geospatial benchmark that can be used to evaluate spatial analysis on big data platforms. This research should progress in several directions. First, we propose to develop a complementary benchmark for vector datasets. Many platforms provide spatial analysis operators for vector spatial data, and this benchmark should be extended to encompass the two major spatial data types. Secondly, work comparing the platforms should be extended into different computation environments. Our work examined large memory single node high-performance environment, but results could differ substantially in a distributed computing environment. Lastly, the evaluation framework should be updated with more platforms, new versions of the existing platforms, larger multi-spectral datasets, and the integration of spatial workflows. The development of such an ambitious geospatial benchmark would be beneficial to the entire geospatial community as it would provide a clear framework that identifies pain-points and successes for high-performance geospatial computing.

References

1. Boshuizen, C., Mason, J., Klupar, P. & Spanhake, S. Results from the planet labs flock constellation. (2014).
2. Yang, C., Huang, Q., Li, Z., Liu, K. & Hu, F. Big Data and cloud computing: innovation opportunities and challenges. *Int. J. Digit. Earth* **10**, 13–53 (2017).
3. Haynes, D. Array Databases. *Geogr. Inf. Sci. Technol. Body Knowl.* **2019**, (2019).
4. Palamuttam, R. *et al.* SciSpark: Applying in-memory distributed computing to weather event detection and tracking. in *2015 IEEE International Conference on Big Data (Big Data)* 2020–2026 (IEEE, 2015). doi:10.1109/BigData.2015.7363983.
5. Wang, W. *et al.* SparkArray: An Array-Based Scientific Data Management System Built on Apache Spark. in *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)* 1–10 (2016). doi:10.1109/NAS.2016.7549422.
6. Li, H. *et al.* FASTDB: An Array Database System for Efficient Storing and Analyzing Massive Scientific Data. in *Algorithms and Architectures for Parallel Processing* (eds. Wang, G., Zomaya, A., Martinez, G. & Li, K.) 606–616 (Springer International Publishing, 2015).
7. Appel, M., Lahn, F., Pebesma, E., Buytaert, W. & Moulds, S. Scalable earth-observation analytics for geoscientists: Spacetime extensions to the array database SciDB. in *EGU General Assembly Conference Abstracts* vol. 18 11780 (2016).
8. Jiang, L., Kawashima, H. & Tatebe, O. Fast window aggregate on array database by recursive incremental computation. in *2016 IEEE 12th International Conference on e-Science (e-Science)* 101–110 (2016). doi:10.1109/eScience.2016.7870890.
9. Lu, M., Appel, M. & Pebesma, E. Multidimensional Arrays for Analysing Geoscientific Data. *ISPRS Int. J. Geo-Inf.* **7**, 313 (2018).
10. Planthaber, G., Stonebraker, M. & Frew, J. EarthDB: scalable analysis of MODIS data using SciDB. in *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data* 11–19 (2012).
11. Karmas, A., Karantzalos, K. & Athanasiou, S. Online analysis of remote sensing data for agricultural applications. in *OSGeo's European Conference on Free and Open Source Software for Geospatial* (2014).
12. Picoli, M. C. A. *et al.* Big earth observation time series analysis for monitoring Brazilian agriculture. *ISPRS J. Photogramm. Remote Sens.* **145**, 328–339 (2018).
13. Sidhu, N., Pebesma, E. & Câmara, G. Using Google Earth Engine to detect land cover change: Singapore as a use case. *Eur. J. Remote Sens.* **51**, 486–500 (2018).
14. Eldawy, A. & Mokbel, M. F. The era of big spatial data. in *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on* 42–49 (2015).
15. Doan, K. *et al.* Evaluating the impact of data placement to spark and SciDB with an Earth Science use case. in *Big Data (Big Data), 2016 IEEE International Conference on* 341–346 (IEEE, 2016).
16. Olasz, A., Thai, B. N. & Kristóf, D. A New Initiative for Tiling, Stitching and Processing Geospatial Big Dat in Distributed Computing Environments. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **3**, (2016).
17. Haynes, D., Manson, S. & Shook, E. Terra Populus' Architecture for Integrated Big Geospatial Services. *Trans. GIS* **21**, 546–559 (2017).
18. Wiener, P., Simko, V. & Nimis, J. Taming the Evolution of Big Data and its Technologies in BigGIS-A Conceptual Architectural Framework for Spatio-Temporal Analytics at Scale. in *GISTAM* 90–101 (2017).
19. Ray, S., Simion, B. & Brown, A. D. Jackpine: A benchmark to evaluate spatial database performance. in *2011 IEEE 27th International Conference on Data Engineering* 1139–1150 (2011). doi:10.1109/ICDE.2011.5767929.
20. Rabl, T. *et al. Big Data Benchmarking: 6th International Workshop, WBDB 2015, Toronto, ON, Canada, June 16-17, 2015 and 7th International Workshop, WBDB 2015, New Delhi, India, December 14-15, 2015, Revised Selected Papers.* vol. 10044 (Springer, 2016).
21. Sharma, M., Paige, G. B. & Miller, S. N. DEM development from ground-based LiDAR data: A method to remove non-surface objects. *Remote Sens.* **2**, 2629–2642 (2010).
22. Ding, Y. & Densham, P. J. Spatial strategies for parallel spatial modelling. *Int. J. Geogr. Inf. Syst.* **10**, 669–698 (1996).
23. Siegel, D., Doney, S. & Yoder, J. The North Atlantic spring phytoplankton bloom and Sverdrup's critical depth hypothesis. *science* **296**, 730–733 (2002).
24. Zhou, P., Luukkanen, O., Tokola, T. & Nieminen, J. Effect of vegetation cover on soil erosion in a mountainous watershed. *Catena* **75**, 319–325 (2008).
25. Stonebraker, M., Brown, P., Poliakov, A. & Raman, S. The architecture of SciDB. in *International Conference on Scientific and Statistical Database Management* 1–16 (Springer, 2011).

26. Camara, G. *et al.* Big earth observation data analytics: matching requirements to system architectures. in *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data - BigSpatial '16* 1–6 (ACM Press, 2016). doi:10.1145/3006386.3006393.
27. Lu, M., Pebesma, E., Sanchez, A. & Verbesselt, J. Spatio-temporal change detection from multidimensional arrays: Detecting deforestation from MODIS time series. *ISPRS J. Photogramm. Remote Sens.* **117**, 227–236 (2016).
28. Baumann, P. *et al.* Big Data Analytics for Earth Sciences: the EarthServer approach. *Int. J. Digit. Earth* **9**, 3–29 (2016).
29. National Institute of Space Research. E-Sensing: Bg Earth Observation Data Analytics for LUCC. *http://esensing.org/* http://esensing.org/ (2014).
30. Gu, L. & Li, H. Memory or time: Performance evaluation for iterative operation on hadoop and spark. in *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on* 721–727 (IEEE, 2013).
31. Towns, J. *et al.* XSEDE: accelerating scientific discovery. *Comput. Sci. Eng.* **16**, 62–74 (2014).
32. Wang, Y., Chen, Z., Cheng, L., Li, M. & Wang, J. Parallel scanline algorithm for rapid rasterization of vector geographic data. *Comput. Geosci.* **59**, 31–40 (2013).
33. Eldawy, A., Niu, L., Haynes, D. & Su, Z. Large Scale Analytics of Vector+Raster Big Spatial Data. in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - SIGSPATIAL'17* 1–4 (ACM Press, 2017). doi:10.1145/3139958.3140042.
34. Yang, H., Dasdan, A., Hsiao, R.-L. & Parker, D. S. Map-reduce-merge: simplified relational data processing on large clusters. in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data - SIGMOD '07* 1029 (ACM Press, 2007). doi:10.1145/1247480.1247602.
35. Afrati, F. N. & Ullman, J. D. Optimizing joins in a map-reduce environment. in *Proceedings of the 13th International Conference on Extending Database Technology - EDBT '10* 99 (ACM Press, 2010). doi:10.1145/1739041.1739056.
36. Yu, J., Zhang, Z. & Sarwat, M. Spatial Data Management in Apache Spark: The GeoSpark Perspective and Beyond. *GeoInformatica* 41 (2018).