A Survey on Spatio-temporal Data Analytics Systems

MD MAHBUB ALAM and LUIS TORGO, Dalhousie University, Canada
ALBERT BIFET, The University of Waikato, New Zealand

Due to the surge of spatio-temporal data volume, the popularity of location-based services and applications, and the importance of extracted knowledge from spatio-temporal data to solve a wide range of real-world problems, a plethora of research and development work has been done in the area of spatial and spatio-temporal data analytics in the past decade. The main goal of existing works was to develop algorithms and technologies to capture, store, manage, analyze, and visualize spatial or spatio-temporal data. The researchers have contributed either by adding spatio-temporal support with existing systems, by developing a new system from scratch, or by implementing algorithms for processing spatio-temporal data. The existing ecosystem of spatial and spatio-temporal data analytics systems can be categorized into three groups, (1) spatial databases (SQL and NoSQL), (2) big spatial data processing infrastructures, and (3) programming languages and GIS software. Since existing surveys mostly investigated infrastructures for processing big spatial data, this survey has explored the whole ecosystem of spatial and spatio-temporal analytics. This survey also portrays the importance and future of spatial and spatio-temporal data analytics.

CCS Concepts: • General and reference → Surveys and overviews; • Information systems → Spatial-temporal systems; Parallel and distributed DBMSs;

Additional Key Words and Phrases: Spatial databases, big spatial infrastructures, GIS software, spatial libraries, spatial, spatio-temporal, trajectory, spatial stream

ACM Reference format:
Md Mahbub Alam, Luis Torgo, and Albert Bifet. 2022. A Survey on Spatio-temporal Data Analytics Systems. ACM Comput. Surv. 54, 10s, Article 219 (November 2022), 38 pages.
https://doi.org/10.1145/3507904

1 INTRODUCTION

Due to the technological advancement of the internet, sensor devices, GPS-enabled devices and the popularity of location-based services (LBS) and applications (such as map services, recommendation systems, navigation systems, location-based social networks, and other applications), a huge volume of geo-referenced data is generated every day, often called big spatial data. However, a significant portion of these data comes with a timestamp (temporal-tag), leading to spatio-temporal data. As a result of the availability of smart mobile devices and the internet, LBS applications and services are part of our daily activities and contributing significantly to this data growth. These

The work of L. Torgo was undertaken, in part, thanks to funding from the Canada Research Chairs program and NSERC.
Authors’ addresses: Md M. Alam and L. Torgo, Dalhousie University, 6299 South St., Halifax, NS, Canada, B3H 4R2; emails: {mahbub.alam, ltorgo}@dal.ca; A. Bifet, The University of Waikato, Knighton Road, Hamilton, New Zealand; email: abifet@waikato.ac.nz.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0360-0300/2022/11-ART219 $15.00
https://doi.org/10.1145/3507904

ACM Computing Surveys, Vol. 54, No. 10s, Article 219. Publication date: November 2022.
data also come from other sources such as vehicles, sensors, satellites, space telescopes, aerial photography, land survey, medical imaging, and more. Therefore, mining information from this huge volume of spatio-temporal data is not only important for popular LBS applications and services, but it is also important for scientific discovery and exploration of a wide range of application domains, such as climate change analysis, earthquake analysis, weather forecasting, urban planning, healthcare, transportation system, agriculture, space exploration, crime data analysis, epidemic analysis, animal migration, oceanography, and many more. In this context, there is a demand for efficient tools and data processing systems to store, manage, analyze, and visualize the high dimensional and heterogeneous big spatio-temporal data.

Research and development of spatial and spatio-temporal database systems have started with traditional relational database systems (RDBMSs). Traditional RDBMSs with a spatial extension (e.g., PostgreSQL/PostGIS [185]) are efficient in a single node computing environment. However, due to the lack of parallelism and the I/O bottleneck, these systems only work well for relatively small datasets. Besides, these systems have limited analysis and visualization capabilities. Therefore, one may question if spatial RDBMSs are significant in this era of big spatial data. Researchers are continuously adding new features to make these systems adaptable in this new era. Researchers have also developed a few parallel and distributed systems by using spatial RDBMSs. Though current spatial RDBMSs are not massively scalable, these systems are scalable enough to solve many real-world problems. Also, there is a huge demand for spatial RDBMSs in a wide range of application domains. Therefore, spatial RDBMSs are still significant in this era of big spatial data.

Along with the limitations mentioned above, traditional RDBMSs also did not have support to store and process semi-structured or unstructured data. Therefore, NoSQL database systems (e.g., MongoDB) [66, 101] have emerged as alternative databases, which are schemaless, highly available, and horizontally scalable. Currently, a few of these systems have limited native support to store and process spatial data. Still, researchers have extended some of these databases to add spatial support. Moreover, several big spatial data processing systems have been developed by utilizing the power of NoSQL databases [124, 148, 150, 175, 202]. Spatial support of current NoSQL databases lack spatial analysis and visualization, and only a few of them support SQL-like query language.

In recent years, a number of data processing systems have emerged to process big spatial and spatio-temporal data. These systems are implemented mainly by extending the MapReduce framework Hadoop [16], Spark [276, 277], and NoSQL database systems [66, 101] to incorporate spatial and temporal data types, partitioning and indexing techniques, geometric operations, and an SQL-like query language. However, a few of them have been developed either from scratch [35, 37, 161] or by extending systems other than Hadoop, Spark, and NoSQL databases [5, 6, 78]. Recently, Python libraries such as DASK [72] and RAPIDS [241] emerged as parallel and distributed platforms for processing big data. These systems can be either spatial [81, 172, 269], discrete spatio-temporal [8, 114], trajectory [27, 148, 282], or spatial stream [55, 157, 220] data processing systems. However, these big data systems have limited analysis and visualization capabilities. Also, a few of them support SQL-like query language, but not as efficiently as spatial RDBMSs.

Due to the heterogeneity and implicit spatial and temporal dependencies of spatio-temporal data, being able to extract and analyze knowledge from these data can be extremely challenging. The data mining, analysis, and visualization support of existing spatial RDBMSs, spatial NoSQL databases, and big spatial data infrastructures, are very limited. Whereas, there is a wide range of libraries and packages available for mining, analyzing, and visualizing spatial, spatio-temporal, and trajectory data in two popular de facto programming languages for data science, R [204] and Python [251]. However, these libraries and packages cannot store and process a large volume of data. Therefore, the utilization of these libraries and packages with RDBMSs, NoSQL databases, and big data infrastructures is essential to fill the gap. On the other hand, GIS software like ArcGIS [89]
and QGIS [200] are leading tools to collect, store, process, and visualize spatial data. Initially, GIS software was developed for a single user with limited DBMS capability. Currently, GIS software like ArcGIS has additional tools to utilize the processing capability of Hadoop and Spark. Moreover, Python and R users can use the functionalities of GIS software within their environments.

Researchers from both academia and industry are working in order to meet the current and future demands for big spatio-temporal analytics. Surveys are always crucial for current and future researchers to know about the state-of-the-art. Surveys also help to choose a data analytics system based on application requirements. However, existing surveys and performance analyses are not up-to-date and mostly summarize infrastructures related to big spatial data. In the meantime, a number of big spatio-temporal, trajectory, and spatial stream data processing systems have emerged. Besides, existing surveys have not considered spatial RDBMSs, GIS software, and spatial support in programming languages. These shortcomings motivate the current paper. This survey categorizes the existing ecosystem of spatio-temporal data analytics (see Figure 1) into three data dimensional groups: (1) Data Storage: RDBMSs and NoSQL databases, by first defining the significance of spatial RDBMSs (e.g., PostgreSQL/PostGIS) in this era of big spatial data, and then reviewing the spatial support of popular NoSQL databases; (2) Data Processing: big data infrastructures, where data processing systems are classified based on underlying architecture (e.g., Hadoop, Spark, NoSQL, and others) and the type of data processing system (such as spatial, spatio-temporal, trajectory, and spatial stream); (3) Data Programming and Software Tools, which summarizes available libraries and packages for processing spatial, spatio-temporal, and trajectory data in widely used programming languages, such as R and Python. Two popular GIS software, ArcGIS and QGIS, are also discussed in this last category.

The rest of the paper is organized as follows. Section 2 provides an overview of existing surveys along with limitations, thus setting the goals of this survey. Section 3 discusses the importance of spatio-temporal data analytics research, along with a few important application domains. Spatial and spatio-temporal data is defined in Section 4. Section 5 defines the significance of spatial RDBMSs in the era of big spatial data and discusses the spatial support of NoSQL systems. The detailed review of existing big data infrastructures for processing spatial and spatio-temporal data is presented in Section 6. Section 7 presents the ecosystem containing libraries and packages of two popular languages, Python and R, along with two popular GIS software, ArcGIS and QGIS. Finally, Section 8 concludes the paper.

2 RELATED WORKS

A number of research works have been published which surveyed existing spatial data analytic systems. These surveys either performed a comparative analysis of existing spatial data analytics based on supported features or evaluated the performance of the existing systems by running
supported spatial queries. Therefore, we divided the existing works into two groups, (1) surveys and (2) performance analyses.

**Existing surveys:** In their survey of big spatial data systems, Eldawy et al. [82] have explored the existing works that are developed on or before 2016, based on six key features. These features include (i) the system implementation approach (on-top, built-in, and from-scratch), (ii) underlying data processing architecture (MapReduce, RDD, Parallel DB, etc.), (iii) query language, (iv) indexing techniques, (v) spatial operations, and (vi) visualization. Maguerria et al. [156] presented a comprehensive review of big spatio-temporal data processing systems in the context of the underlying processing frameworks, partitioning and indexing techniques, and supported spatial queries. Castro et al. [54] analyzed the supported spatial features of Hadoop and Spark-based systems from the user’s viewpoint to help users to select spatial data processing systems for their applications. Yao et al. [266] studied and discussed recent technologies and techniques for big spatial vector data management based on the data model, storage, indexing, and processing and analysis. Karim et al. [134] portrayed the spatio-temporal aspect of big data and performed a comparison of the supported spatio-temporal features on different frameworks, such as Hadoop [16], Spark [276], Samza [176], Storm [15], and Flink [95]. Almeida et al. [67] presented a survey on big trajectory data analytics, which provides an overview of a few systems for processing big trajectory data along with traditional systems based on PostgreSQL/PostGIS, Oracle Spatial, and other databases. The big trajectory systems include a cloud-based system on Microsoft Azure [29], ST-Hadoop [8], TrajSpark [282], DiStRDF [174], and systems based on Flink, MongoDB [127], and other databases for semantic trajectories. Recently, Guo et al. [111] have surveyed the geospatial data processing capabilities in the 10 most popular NoSQL databases based on supported geometry types, geometry functions, spatial indexes, query languages, and data formats.

**Existing performance analyses:** Hulbert et al. [125] performed an experimental study on GeoMesa [152] and Elasticsearch [52] by running spatio-temporal queries, where the authors compared these systems based on query execution time and throughput. García et al. [97] conducted a comparative analysis of the performance of SpatialHadoop [81] and LocationSpark [235] based on parallel and distributed spatial distance join queries. Hagedorn et al. [115] performed feature comparison and performance analysis of a few Hadoop and Spark-based spatial data systems. They have conducted a performance evaluation by running range and spatial join queries on SpatialHadoop, SpatialSpark [268], GeoSpark [269], and STARK [114], where STARK is proposed by the authors as a spatio-temporal extension of Spark. Data Reply [209] published a report on benchmarking six big geospatial data infrastructures that include GeoSpark, Hive [247], MongoDB, GeoMesa, Elasticsearch, and Postgres-XL [239]. These systems are evaluated by running several queries (range, regular expression, and join queries) on three different datasets. Pandey et al. [187] performed a comprehensive study to analyze available features of selected Hadoop and Spark-based spatial data processing systems. They also evaluated the performance of five Spark-based spatial data systems (SpatialSpark, GeoSpark, Magellan [229], LocationSpark, and Simba [264]) based on supported spatial operations, which include range and kNN query, spatial join, distance join, and kNN join. Alam et al. [5] also performed a comprehensive feature analysis and performance evaluation on Hadoop and Spark-based spatial data systems. However, instead of evaluating limited supported spatial features, they have implemented OGC-compliant join predicates and analysis features on SpatialHadoop and GeoSpark to assess the performance by running a number of spatial join, spatial analysis, and range queries. The authors also included SpatialIgnite as part of the evaluation, which is developed as extended spatial support for another in-memory computing system, Apache Ignite [13]. More et al. [186] performed an experimental study on SpatialHadoop and GeoSpark to evaluate the performance of data compression, indexing, and kNN and range query on a single node computer, which is not an ideal scenario for evaluating big spatial
data systems. Recently, Haynes et al. [118] proposed a benchmark to evaluate the performance of five raster operations on three big data platforms, namely PostgreSQL/PostGIS [185], SciDB [37], and GeoTrellis [76], using three different datasets. These operations include pixel count, reclassification, raster add, focal operations, and zonal statistics.

**Limitations of existing surveys and performance analyses:** First, existing surveys and performance analyses are not up-to-date. These researches mostly covered systems that were developed on or before 2017. Second, most of these works analyze the big data infrastructures for processing spatial data. However, in the meantime, a number of big data systems for processing spatio-temporal, trajectory, and spatial data streams have emerged. Though Guo et al. [111] have reviewed native spatial supports of NoSQL databases, there is no comprehensive survey on big spatial data systems, which are developed by utilizing NoSQL databases. Moreover, there is no review available on Python libraries such as DASK and RAPIDS for big spatial data processing. However, at present, along with Spark, these parallel and distributed Python libraries are gaining popularity and often considered as the next big data processing platform. Finally, researchers have not considered the other parts of the ecosystem of spatio-temporal analytics, namely spatial RDBMSs, GIS software, and spatial support in programming languages.

**Goal of this survey:** The goal of this survey is to conduct a comprehensive review on the current state of spatio-temporal data analytics systems research for processing spatial, spatio-temporal, and trajectory data. This survey discusses the up-to-date spatial support in RDBMSs, NoSQL databases, big data processing platforms, programming languages, and GIS software. We have studied and discussed major big data platforms based on the type of supports, such as spatial (vector, raster), spatio-temporal, trajectory, and spatial streams. This survey also includes two new emerging big data processing platforms, DASK and RAPIDS. Till now, there was no survey on spatial support in programming languages. This survey provides an overview of available spatial libraries and packages of R and Python. The APIs for interfacing R and Python with spatial RDBMSs, GIS software, and big data platforms are also discussed. Besides, a summary of spatial supports in other popular programming languages (e.g., C/C++, Java) is provided. Finally, this survey presents a review of two popular GIS software, ArcGIS and QGIS. We hope that this survey will help researchers, developers, and other stakeholders towards furthering the state-of-the-art.

### 3 IMPORTANCE AND APPLICATIONS OF SPATIO-TEMPORAL DATA ANALYTICS

In 1854, nearly 10 percent of a neighborhood at 40 Broad Street of the city of London died in just seven days due to the severe outbreak of the deadly disease Cholera. Dr. John Snow [227], a British Physician, was able to identify the source of the disease by plotting cases of Cholera on a map called ghost map [130]. This deadly map has taught us that the inherent knowledge of a map can solve a problem. Now, if we look at the current coronavirus (COVID-19) situation, or last year’s bushfire situation in Australia, or the aftermath of recent floods/hurricanes around the world, or the effects of climate change, we observe that a large volume of people were affected by these deadly events. Forests were burnt, and animals lost their habitats. Though we are more technologically capable than ever, we are facing these situations or problems with more strength and more frequently. Therefore, it is essential to utilize the knowledge of spatial and temporal properties of data to mitigate or tackle many of these problems. Researchers have been using these data for urban planning, navigating vehicles, identifying road accidents, tracking the activity of diseases (e.g., flu) and natural phenomena (e.g., hurricanes, tornados), and solving many other problems. In this section, we discuss some important application domains, which need special attention now and in the future.

The ongoing COVID-19 virus situation came as a shock and has stopped the pace of the world. People lost their jobs and businesses, healthcare systems are overwhelmed with patients, many
countries are struggling to give minimum services to emergency patients, and governments are facing difficulties to maintain supplies of food and necessary medical equipment. Spatio-temporal data analysis, visualization, and mapping in the domain of epidemiology and public health may become essential tools for tackling future pandemics. A number of research works have discovered spatio-temporal patterns \[160\] and the spread of the diseases by studying the patient’s treatment history. However, along with technological support, there is a need to utilize the spatio-temporal tools and techniques for efficient studies of the pandemics to quickly find the originating point, to stop spreading and isolating patients, and to provide better health-care services.

Similarly, the 2019–2020 season bushfire in Australia was the biggest in Australian history \[9\]. As of March 2020, the bushfire burnt more than 18 million hectares of land, destroyed over 5,900 buildings, killed at least 34 people and 3 billion animals, and some animals may be driven to extinction \[51, 248\]. According to NASA, 306 million tonnes of CO\(_2\) were emitted during the 2019–2020 Australian bushfire season \[144\]. Besides, countries like USA, Canada, and a few European countries are also affected by bushfires every year. Spatio-temporal analysis of bushfires based on aerial and satellite images is useful to spot and tackle the bushfires at the initial stage. Also, scientists have warned that if it is not possible to control the emission of greenhouse gas, bushfires could become a normal scenario every year \[107\]. On the other hand, due to cyclones, hurricanes, and floods, a huge volume of people were affected and died around the world every year. Thus, climatology is another pivotal field to discover spatio-temporal patterns and relationships of climate variables and helps to prepare to tackle future adverse conditions \[25\]. Since pollution (air, water, sound, and other pollution) is an ongoing issue for a long time, the study of the environmental science domain to discover factors of pollution using data collected from sensors will always be as crucial as today.

A large number of people are affected by an increasing number of large scale crises and disasters every year, such as hurricanes, bushfires, floods, earthquakes, epidemics, and other events. These also can be small scale local emergency events such as road accidents, crimes, and house fires. During any such events, immediate actions are required to mitigate the suffering of people. These actions include to rescue and alert peoples, to maintain the supply chains (foods, medical supplies, and other resources), to provide medical services, and other necessary steps. Currently, GIS tools and solutions \[1, 39, 40, 217\] are used by emergency teams around the world to analyze data collected from aerial drones, satellites, smartphones, social networks, and other sources to take immediate measures. Future geospatial analytics will be more advanced and accurate for emergency management and response due to the integration of AI and machine learning.

As the ocean and marine environment is the largest part of the earth, spatio-temporal ocean and marine datasets collected from widespread sources (such as satellites, remote sensors, aerial drones, stations, ships, buoys, and underwater sensors) are valuable in many application domains. These domains include safe and secure maritime navigation, autonomous cargo shipping, aquaculture production optimization, improved detection and forecasting of environmental changes, advanced weather forecasting, anomaly detection for identifying smuggling or drug trafficking, maritime surveillance, classifying acoustic sounds, and more. In addition, maritime shipping is the backbone of world trade and manufacturing supply chains. Therefore, application domains related to the ocean and marine environment are hot fields of research \[262\].

Due to the advancement of GPS technology and the internet, most taxis are equipped with a GPS device in large cities, and people use online location-based services, such as Uber, Google Maps, and other services for traveling purposes. A taxi driver always wants to get a passenger quickly and maximize profit, whereas a passenger wants to reach a destination as quickly as possible. A number of research works \[98, 203, 272, 274\] have been done in the last couple of years intending to maximize the profit of taxi drivers and utilize the valuable time of passengers. The outcome of
these researches was already implemented into location-based services we are using today. This research also helps traffic engineers to implement policies to reduce traffic congestion and identify road accidents. However, as we are moving towards driver-less taxi services, intelligent transportation systems is becoming an important field of research. On the other hand, researchers also started utilizing data generated from urban areas (such as sensors, vehicles, and humans) for urban design and planning \[271, 273\], which is another domain to watch-out.

We need to increase agricultural production to ensure food security for the growing population of the world. However, the arable land area is decreasing due to the high growth of population and urbanization. Also, the fertility of lands is reducing as a result of the excessive use of fertilizers, pesticides, herbicides, water, and other inputs. Besides, crop production is affected by floods, drought, soil erosion, and other calamities. On top of that, farmers might lose farmlands due to the rise of sea levels as an effect of climate change. Increasing crop production as well as reducing the cost of production and unnecessary use of farm inputs are very challenging tasks. Precision agriculture \[70, 86, 108\] uses knowledge extracted from spatio-temporal data collected from aerial drones, satellites, sensors, and other sources to identify soil types, crop diseases, and other attributes. This knowledge helps farmers to identify site-specific needs and to optimize the use of farm inputs to maximize crop production and profits.

Other application domains, such as Animal Migration, Space Exploration, and Neuroscience, are also prominent fields of research. Researchers also suggested exploring beyond the traditional application domains, such as biology, chemistry, astronomy, and more \[255\].

4 DEFINITION AND TYPES OF SPATIAL AND SPATIO-TEMPORAL DATA

Currently available spatial databases, big spatial data infrastructures, programming languages, and software tools have built support to model, store, and process either spatial or spatio-temporal data. Spatio-temporal data can be either discrete point data or trajectory data. This section will define and differentiate among these types of data.

A data item related to space (location-aware or geo-tagged) is called spatial data. Traditionally, raster data (e.g., satellite images), point data (e.g., crime reports), or network data (e.g., road maps) were known patterns of spatial data \[91\]. In recent years, the traditional pattern of spatial data has changed due to the wide adoption of GPS-enabled devices and the popularity of location-based services (LBS) and applications. Examples of this change include check-ins, GPS trajectories of smartphones, geo-tagged tweets, Instagram photos, and so on. Spatial data types can be divided into three categories: vector, raster, and network data. Raster data is represented as a collection of pixels (or grid cells), where each pixel is associated with a specific geographical location. Raster data can be discrete (such as land-cover type, soil type) or continuous (such as temperature, elevation, satellite images). Vector data can be represented by points (e.g., a city, a movie theater), lines (e.g., roads, rivers) or polygons (e.g., a lake, a national park). A spatial network is a special graph that consists of nodes embedded in space. The most common example of a spatial network is the transportation network (e.g., the road network), where edges represent road segments, and nodes represent the intersection of road segments or points of interest \[123\].

On the other hand, spatial data is being captured with a timestamp (temporal-tag) called spatio-temporal data, i.e., spatio-temporal data contains both spatial and temporal aspects of an object. Spatio-temporal data can also be defined as geometries changing over time \[85\]. There are a number of data models (such as event model, temporal snapshot model, temporal change model and more) to represent spatio-temporal data in data processing systems \[197, 223\]. Spatio-temporal data type is basically the integration of timestamps (e.g., time instance, period, interval) with the spatial data type (e.g., point, line, polygon). Several classes of spatio-temporal data types are available in real-life application domains to represent an object with respect to both space and time.
Kisilevich et al. [136] have considered point objects and defined five classes of spatio-temporal types, which include spatio-temporal events, geo-referenced variables, geo-referenced time series, moving objects, and trajectories. Whereas, Atluri et al. [25] have described four classes of spatio-temporal data types, such as event data, point reference data, trajectory data, and raster data.

When raster data is collected with a timestamp, it is called spatio-temporal raster data. For example, air quality observations data from ground-based sensors or earth surface observations data from satellites are raster spatio-temporal data [25]. In both of these cases, data is collected at fixed locations in space over time. The classification of vector data with timestamps depends on the type of geometries, such as points, lines, and polygons [85]. If we consider spatial points and point of time, spatio-temporal data can be either discrete point data or trajectory data. Spatio-temporal discrete point data can be event data that represent where and when the event happened. For example, a traffic accident can be represented by accident location and time of the accident. Therefore, event data can be used to model many real-life events such as disease outbreaks, plane crashes, volcano eruptions, and more [25]. Spatio-temporal discrete point data can also be point reference data, which is collected from a set of moving reference points in space over time. For example, drone observations of bushfires at point locations in space over time [25]. Whereas, a trajectory is a path that consists of a set of points generated by moving objects in geographical space over time. The main sources of trajectory data are either GPS-enabled devices (e.g., taxi trajectories) or sensors attached to moving objects (e.g., animal trajectories). Trajectories can be classified into four main categories, such as mobility of humans, vehicles, animals, and natural phenomena (such as hurricanes, tornados) [283]. The knowledge derived from trajectory data is important in many application domains, such as intelligent transportation systems, urban planning, recommendation systems, animal migration analysis, and more.

Similarly, spatio-temporal data can be classified by considering other geometries like lines and polygons instead of points, and timestamps like interval and period instead of a point of time [85, 136]. Finally, due to the nature of spatio-temporal data, we can perform queries based on spatial, temporal, and spatio-temporal properties and relationships.

5 SPATIAL DATABASES

Spatial databases can be divided into two main categories, relational databases (SQL), and NoSQL databases. Traditional RDBMSs with spatial support are stable, mature, efficient and have been used in a wide range of application domains. Due to the large volume and diverse form of data being generated from a wide range of sources, NoSQL database systems, and big data processing platforms have emerged. One may ask what is the significance of spatial RDBMSs in this era of big spatial data. However, spatial RDBMSs are adapting to this era by integrating new features continuously. Researchers from industry and academia also developed a few parallel and distributed systems by utilizing spatial RDBMSs. Therefore, these modern spatial RDBMSs are still significant in a wide range of application domains. This section will address the significance of modern spatial RDBMSs in this era of big spatial data and explores the spatial support of both SQL and NoSQL databases.

5.1 Spatial Relational Databases

Traditional RDBMSs are popular for efficient data management and query processing. Therefore, research and development of spatial and spatio-temporal database systems have started by adding extensions to traditional RDBMSs. For example, PostgreSQL/PostGIS [185], Oracle Spatial [182], IBM DB2 Spatial Extender [2], Microsoft SQL Server [92], MySQL Spatial [168], and SpatialLite [96] are some popular spatial RDBMSs. The up-to-date features of these spatial RDBMSs are summarized in Table 1. These RDBMSs have an efficient SQL query engine and support common data
Table 1. Popular Spatial RDBMSs

| RDBMS               | Data Formats                  | Geometry Types                        | Spatial Indexing | Raster Support | Spatial Functions          |
|---------------------|-------------------------------|---------------------------------------|------------------|----------------|-----------------------------|
| PostgreSQL/PostGIS  | WKT, WKB, GML, KML, GeoJSON, | Point, LineString, Polygon, Collections | GIST, SP-GIST,   | Yes            | OGC SFA-SQL, ISO SQL/MM     |
|                     | SVG                           |                                       | BRIN             |                |                             |
| Oracle Spatial      | WKT, WKB, JSON, GeoJSON       | Point, LineString, Polygon, Collections | R-Tree           | Yes            | OGC SFA-SQL, ISO SQL/MM     |
| Microsoft SQL Server| WKT, WKB, GML, GeoJSON        | Point, LineString, Polygon, Collections | Multi-level Grid | No             | OGC SFA-SQL, ISO SQL/MM     |
| IBM DB2 Spatial Extender | WKT, GML, KML, ESRI Shapefile | Point, LineString, Polygon, Collections | Spatial Grid     | No             | OGC SFA-SQL, ISO SQL/MM     |
| MySQL Spatial      | WKT, WKB                      | Point, LineString, Polygon, Collections | R-Tree           | No             | OGC SFA-SQL                 |
| SQLite/SpatialLite  | WKT, WKB                      | Point, LineString, Polygon, Collections | R*-Tree          | No             | OGC SFA-SQL                 |

1 Collections - MultiPoint, MultiLineString, MultiPolygon, GeometryCollection.
2 Support functions compliant with OGC SFA-SQL (Part-2) [177] and ISO SQL/MM (Part-3) [230] standard.

formats (WKT, WKB) and geometry objects (point, linestring, polygon) specified by OGC [177]. Also, most of these spatial RDBMSs support R-Tree type indexing, except SQL Server and IBM DB2, where grid indexing has been utilized. Among them, only PostgreSQL/PostGIS and Oracle Spatial can store and process spatial raster data. Popular spatial RDBMSs, such as PostgreSQL/PostGIS, Oracle Spatial, and SQL Server, are fully compliant with OGC [177] and ISO SQL/MM (part-3) [230]. Therefore, a wide range of spatial queries (e.g., spatial join, range) can be executed in these databases.

However, due to the I/O bottleneck, lack of parallelism and scalability, the performance of these systems deteriorated with the increasing volume of data. Also, it is challenging to model heterogeneous and multidimensional data in spatial RDBMSs. Still, these databases went through a lot of changes in the last couple of years. Researchers and developers are continuously integrating new features to these systems or utilizing these systems to meet the current demands of spatial data analysis. Therefore, this section will also address the significance of modern spatial RDBMSs in this era of big spatial data by discussing changes made in one of the most popular spatial RDBMS, PostgreSQL/PostGIS.

PostgreSQL is an open-source, vertically scalable, and extensible RDBMS. PostGIS is a spatial extension of PostgreSQL, which supports OGC-compliant spatial SQL queries. Vertical scaling can improve the performance of PostgreSQL/PostGIS on a single computer system, but horizontal scalability is required for processing a large volume of spatial data. We can achieve horizontal scalability through sharding in PostgreSQL. Sharding can reduce the I/O bottleneck significantly by partitioning data across the cluster [164]. Several solutions are available where horizontal scalability and query parallelism is achieved through sharding, such as Postgres-XL [239], Citus [65], PL/Proxy [238], etc. PostGIS can be integrated with both Citus and Postgres-XL. PostgreSQL (v9.6+) also has a built-in sharding feature called Foreign Data Wrappers (FDW) that allows PostgreSQL to access data from external sources instead of local tables. Moreover, features like parallel sequence scans, parallel joins, and parallel aggregates for parallel spatial query processing are now completely working with PostgreSQL (v12) and PostGIS (v3.0). Thus, one can take advantage of default parallel processing support to process large scale spatial data in PostgreSQL/PostGIS [205].

As today’s big data comes from diverse sources in different formats, it is not always possible to store these data in a tabular format in RDBMS. Therefore, NoSQL database systems emerged in the last decade. However, JSON and JSONB data types were added to PostgreSQL in 2012 and
2014, respectively. Recently, SQL/JSON was introduced in PostgreSQL v12, which is compliant with the SQL-2016 standard. The SQL-2016 standard has recognized NoSQL and includes features for the SQL/JSON data model and path language as well as commands for storing, publishing, and querying JSON data. Thus, now we can query and index JSON data in PostgreSQL [138].

Some research work has also utilized PostgreSQL/PostGIS to develop parallel query processing infrastructures for spatial data [4]. For example, Niharika [206] utilizes the powerful features of PostgreSQL/PostGIS to implement a parallel query processing system along with efficient data declustering and load balancing techniques. However, as its storage layer is not distributed, it needs to replicate the whole dataset in each node of a cluster. On the contrary, each node of Paragon [119] needs to host a subset of the partitions only. MobilityDB [284] was developed as an extension of PostgreSQL/PostGIS, which provides support for storing and querying moving objects data (trajectory). This support includes spatio-temporal data types, indexing techniques, and query operations. Recently, MobilityDB emerged as a distributed system by integrating with Citus [65] for processing massive trajectory data [28]. The spatial data analysis platform CARTO [53] also uses PostgreSQL/PostGIS underneath as a spatial database server.

Similarly, other spatial RDBMSs (e.g., Oracle Spatial and SQL Server) are also incorporating new features continuously to adapt to this era of big spatial data. Besides, a large number of companies are still using spatial RDBMSs for their businesses. This means that currently, if someone ask the following questions: (i) can spatial RDBMSs (like PostgreSQL/PostGIS) scale well for the problems we are dealing today?; (ii) can these systems process massive datasets?; (iii) can these systems process data in different formats?; or (iv) how long will these systems survive in this era of big data?: the answers to these questions would be yes, these systems are scalable enough for many problems we are dealing with today and can process a certain volume of data in different formats, and will be around for a long time.

5.2 Spatial NoSQL Databases

NoSQL (Not-Only-SQL) database systems [66, 101] are also known as non-relational database systems. Since a large volume of data comes from diverse sources with various formats (such as semi-structured and unstructured), it is challenging to model these data using relational tables. Besides, traditional relational database systems suffer due to the lack of parallelism, I/O bottleneck, and horizontal scalability. Therefore, NoSQL systems have emerged as schema-free, fault-tolerant, scalable, and highly available database systems in the last decade. NoSQL database systems can be classified into four broad groups based on the core data model, (1) Key-Value Databases (e.g., Redis, Oracle NoSQL), (2) Column Family (Wide-Column) Databases (e.g., Cassandra, HBase), (3) Document Databases (e.g., MongoDB, Couchbase), and (4) Graph Databases (e.g., Neo4j, ArangoDB).

Only a few of the NoSQL systems have native spatial support currently. Spatial extensions have been added to some of the NoSQL databases in recent years. The current spatial support of some popular NoSQL databases is presented in Table 2.

Redis [207] is an in-memory key-value store that has implemented a geohash spatial index to accelerate query processing. A set of commands (e.g., geoadd, geopos, geohash, georadius, etc.) are available in Redis to create an index and perform spatial operations on point datasets stored in Geo Sets. However, these spatial commands can only perform limited spatial analyses on point type geometry. Also, Redis does not support SQL-like query language. On the contrary, one can run SQL-like spatial queries on Oracle NoSQL [180], which supports all common geometry objects, geohash indexing, and a set of spatial operators for processing spatial data.

MongoDB [127] is a document database that has native support for processing spatial data. MongoDB supports common GeoJSON objects (such as point, linestring, and polygon) and $2dsphere indexes to model geometries on a spherical surface. It can also store geometries on a 2D surface...
| Data Model       | Spatial Support   | Data Formats     | Geometry Types | Spatial Indexing | Spatial Functions                  | SQL-like Query |
|------------------|-------------------|------------------|----------------|------------------|------------------------------------|----------------|
| Redis            | Key-Value         | Native [208]     | GeoJSON        | Point            | geohash                            | Not Supported  |
| Oracle NoSQL     | Key-Value         | Native [180]     | GeoJSON        | Point, LineString, Polygon, Collections¹ | geohash, geo_intersect, geo_inside, geo_near, geo_within_distance | Yes Supported  |
| MongoDB          | Document          | Native [127]     | GeoJSON, Legacy Coord Pairs | Point, LineString, Polygon, Collections | 2dsphere, 2d | $near, $SnearSphere, $geoWithin, $geoNear, $geoIntersect | Not Supported |
| Couchbase        | Document          | GeoCouch [61]    | GeoJSON        | Point, LineString, Polygon, Collections | R-Tree | BBox | N1QL |
| Cassandra        | Wide_Column       | Lucene Index Plugin [129] | WKT            | Point, LineString, Polygon, Collections | Lucene Index (Secondary Index) | intersects, contains, is_within | CQL |
| Neo4j            | Graph             | Native           | N/A            | Point (2D, 3D)   | Hilbert-curve                      | Cypher |
|                  | Neo4j-Spatial [169] | WKT, WKB        | Point, LineString, Polygon, Collections | R-Tree | Contains, Cover, Cross, Overlap, Touch, Within, Disjoint, Intersect, etc. |

¹Collections - MultiPoint, MultiLineString, MultiPolygon, GeometryCollection.

as legacy coordinate pairs and 2d indexes to model 2D queries. MongoDB provides a set of operators such as $Snear, $SnearSphere, $SgeoWithin, $SgeoIntersect, and $geoNear to perform spatial queries. However, like Redis, it does not have support for SQL-like queries. Whereas, Couchbase [236] supports SQL-like query language, N1QL. GeoCouch [61] is a spatial extension for both Couchbase and CouchDB [19]. GeoCouch is developed based on R-Trees and supports common GeoJSON objects like MongoDB. It allows executing spatial queries using bounding-boxes (BBox). However, MongoDB is richer in terms of support to perform a wide range of spatial queries.

Cassandra [14] is a column family database that does not have native support for processing spatial data. Stratio’s Lucene Index [129] is a spatial plugin for Cassandra, whose spatial index is an extension of Cassandra’s secondary indexes. The Lucene plugin provides a set of spatial predicates (intersects, contains, and is_within) and transformation functions (buffer, convex hull, union, and more), which enable Cassandra to store, index, and process common spatial objects such as point, linestring, polygon, and collections. Brahim et al. [46] have also extended CQL (Cassandra Query Language) to add spatial support with Cassandra, which includes geohash indexing and spatial queries (within_circle, within_polygon, and within_path).

Neo4j [128] is one of the most popular graph database systems, which supports an efficient query language, Cypher. Neo4j Spatial [169] is a library that facilitates Neo4j to store, index, and process spatial data. This library contains modules to import spatial data (ESRIShapefile and OSM), and R-Tree indexing can be applied during import or later to stored data. It also supports a wide range of spatial functions (contain, cover, intersect, and so on) to perform spatial operations on common geometric objects (point, linestring, polygon, and collections). Besides, it wraps popular geospatial libraries, JTS [153] and GeoTools, and therefore, one can utilize the functionalities of these libraries in Neo4j. However, Neo4j spatial is an external library, and hence, it is not highly scalable. Moreover, this library suffers when applications require high concurrency and need to handle a large volume of data. Therefore, Neo4j (v3.4) introduced two native data types, spatial (Point) and temporal (Date, Time, DateTime, Duration, and other types). This point type supports both 2D and 3D points and can be specified by either a geographic or cartesian coordinate system. Neo4j uses Hilbert-curve for indexing points (2D or 3D) and only supports spatial distance function.

Researchers have also developed several big spatial data processing systems by utilizing the capability of NoSQL databases in recent years that will be discussed in Section 6.3. The performance of NoSQL databases is evaluated and discussed by several researchers for spatial
workloads [133, 135]. Some of these performance analyses also involve comparisons with relational spatial databases [30, 31, 131, 159].

5.3 Future Research Directions

Considering the current state of spatial RDBMSs, it would be a great idea to incorporate distributed in-memory or disk-based storage with parallel and distributed systems developed based on spatial RDBMSs. Currently, the spatial support of NoSQL databases lacks available spatial operations compared to spatial RDBMSs. Also, a few of these databases do not have support for SQL-like spatial queries. In addition, we need to work to add support to store and process spatial raster and trajectory data in NoSQL databases. At present, the graph database, Neo4j (v4.0), can scale horizontally through sharding, and therefore, it will be interesting to see the performance of distributed graph databases like Neo4j for processing spatial data.

6 BIG SPATIO-TEMPORAL DATA PROCESSING INFRASTRUCTURES

With the rise of big spatial and spatio-temporal data and its application domains, there is demand for a highly scalable and distributed data processing system. Researchers from both academia and industry are working towards achieving this demand. The big spatial data processing systems that have been developed in the last couple of years are mainly based on MapReduce framework Hadoop [16], NoSQL databases [66, 101], and Spark [276, 277]. Most of these systems have built spatial or spatio-temporal support either by adding a layer on top existing systems or by extending the core of the existing systems. Many of these systems have been developed either from scratch or by utilizing platforms apart from Hadoop, NoSQL, and Spark. In this section, we first categorize the big spatio-temporal data infrastructures based on their development criteria. The systems under each of the categories are then divided into groups based on the type of data processing systems, which include spatial (vector, raster), spatio-temporal, trajectory, and spatial stream. An overview of these systems is provided in Figure 2. Finally, this section provides a comprehensive review of each of these systems based on their supported features, such as data types, partitioning and indexing techniques, query language, and queries.
Hadoop [16] is a highly scalable and distributed MapReduce [68] framework for processing big data. Hadoop does not have any native support for processing spatial (or spatio-temporal) data. Therefore, Hadoop distributes and indexes data across the cluster without considering the spatial (or spatio-temporal) aspect of data, which affects query processing performance on the data negatively. Due to the huge popularity of Hadoop as a big data processing framework in both research and industry communities, a number of extensions to Hadoop were proposed to store, process, and analyze spatial (or spatio-temporal) data. These systems include Hadoop-GIS [3], SpatialHadoop [81], ESRI Tools for Hadoop [259], Parallel SECONDO [113], ST-Hadoop [8], Summit [7], and HadoopTrajectory [27]. A detailed feature matrix of these systems is provided in Table 3.

Hadoop-GIS [3] is a spatial extension of Hadoop. It integrates a spatial layer on top of Hadoop instead of changing the core of the framework. As a result, the performance of Hadoop-GIS for processing spatial data is not quite as good as expected. Besides, Hadoop-GIS extends Hive [247] to support declarative spatial querying (HiveSP) that adds an extra layer of overhead over Hadoop for processing spatial queries. SpatialHadoop [81] incorporates spatial support inside the core of the Hadoop framework. Therefore, it achieves better performance than Hadoop-GIS for running spatial queries on a large dataset. An SQL-like query language, Pigeon [80], which extends Pig Latin [178], is also introduced to run spatial queries on SpatialHadoop.

Due to the lack of spatio-temporal data types, partitioning, and indexing techniques, both Hadoop-GIS and SpatialHadoop suffer when querying on spatio-temporal datasets. ST-Hadoop [8] is a temporal extension of SpatialHadoop, which incorporates spatio-temporal awareness into each layer of SpatialHadoop. However, ST-Hadoop was developed by considering attributes of discrete spatio-temporal point data, not trajectory data, and properties of trajectory data are quite different from discrete point data. Therefore, if we partition and index trajectory data using ST-Hadoop, the performance of query processing will be impacted negatively. For example, each individual trajectory of an object contains a set of points and an object can have multiple trajectories. Now, if we partition trajectories of a moving object using ST-Hadoop, they may be partitioned into different blocks of HDFS over different clusters, which will require more time to perform queries. Therefore, Summit [7] was developed as an extension of ST-Hadoop to include data types, partitioning and indexing techniques, and operations, for processing trajectory data. Bakli et al. [27] have proposed HadoopTrajectory, which adds a diverse set of data types and operators into the core of Hadoop to store and process trajectory data. The careful integration of partitioning and indexing strategies for trajectory data into Hadoop layers makes their system an efficient big trajectory processing system.

### 6.1 Hadoop-based Big Spatio-temporal Infrastructures

| System Type      | Data Types                  | Partitioning | Indexing                  | SQL-Like Query Language | Supported Queries |
|------------------|-----------------------------|--------------|---------------------------|-------------------------|-------------------|
| SpatialHadoop     | Point, LineString, Polygon  | SATO         | Two-Level (Global, Local) | R*-tree                 | Range, kNN, Join  |
| Parallel SECONDO | Point, LineString, Polygon  | SATO         | Two-Level (Global, Local) | R*-tree                 | Range, kNN, Join  |
| Summit            | STPoint, Time, Interval     | Time-Slice   | Two-Level (Temporal, Local) | Spatial: SpatialHadoop [81] | Range, Join       |
| HadoopTrajectory  | Point, Region, Instant, Interval, Periods, TrajSegment, Trajectory | N/A          | Grid, R-Tree (3D Extension) e.g., 3DR-Tree | N/A | Range, kNN, kNN Similarity, Join |

1SATO supports Fixed-Grid, Binary-Space, Hilbert-Curve, Strip-based, and STR partitioning techniques.
system. Parallel SECONDO [113] integrates moving object database SECONDO [112] with Hadoop for scalability. However, Hadoop only does the scheduling and query coordination tasks received from the SECONDO master node, while SECONDO executes the query in each worker node. SECONDO master node is also responsible for aggregating the results. Therefore, parallel SECONDO has not been able to utilize the power of Hadoop properly due to its centralized behavior.

6.2 Spark-based Big Spatio-temporal Infrastructures

Hadoop is a disk-based system optimized for I/O efficiency. Therefore, the performance of Hadoop-based spatial data systems can deteriorate at scale. On the other hand, the growing main memory capacity in a cluster of machines has fueled the development of in-memory big data systems. Spark [276, 277] is a popular and widely used distributed in-memory big data processing framework, which was developed to achieve better performance than disk-based systems. However, like Hadoop, Spark also does not have native support for processing spatial (or spatio-temporal) data. Therefore, several Spark-based spatial (or spatio-temporal) data processing systems have been developed in the last few years. These systems include SpatialSpark [268], GeoSpark [269], LocationSpark [234, 235], Simba [264], STARK [114], SparkGIS [26], TrajSpark [282], Elcano [84], DiStRDF [174], DITA [221], UfTRAMan [75], Dragoon [93], Beast [79], and GeoTrellis [76]. The detailed feature matrix of these systems is presented in Table 4.

GeoSpark [269, 270] is a spatial extension of Spark, which extends Spark RDDs (Resilient Distributed Datasets) [275] to support spatial vector data types called Spatial RDD. It supports several spatial partitioning (Fixed-Grid, Voronoi, R-Tree, and Quad-Tree) and indexing (R-Tree and Quad-Tree) techniques to speed-up spatial queries (range, kNN, and join) on Spatial RDDs. Initially, GeoSpark did not have any support for SQL-like queries [269]. Later, it has introduced an SQL API (SQL/MM-Part 3 Standard) [270] as a spatial extension of Spark SQL [24]. Recently,
GeoSpark was released as Apache Sedona [22] under the Apache Software License. Apache Sedona has also incorporated support for processing spatial raster data and released API for Python and R users [22]. SpatialSpark [268] can perform range and spatial join (broadcast and partitioned) queries over geometric objects. Data can be partitioned using Fixed-Grid, Binary-Split, and Sort-Tile techniques and indexing using R-tree. However, it does not have support for SQL queries. Since SpatialSpark has implemented as a library on top of Spark instead of modifying the core of the framework, it may affect the query performance. Besides, both GeoSpark and SpatialSpark do not have any support for handling data and query skew.

On the other hand, Simba [263, 264] extends Spark SQL [24] and DataFrame API to make spatial support for Spark. It improves the query performance by introducing multi-level (global and local) R-tree indexing on RDDs, and spatial-aware (logical and cost-based) query planning. Moreover, STR partitioner [146] mitigates the data partitioning skew significantly. Later, Simba extended its supports for spatial partitioning and indexing (Table 4). However, Simba only supports spatial operations over point and rectangle objects. LocationSpark [234, 235] was developed as a spatial library like SpatialSpark. It stores spatial data as a key-value pair, where the key can be any geometric object (points, lines, polygons), and the values can be any user-specified text. Like Simba, it also contains an efficient cost model and a query execution planner to deal with data partitioning and query skew. Moreover, along with multi-level indexing (global and local), LocationSpark has introduced a spatial bloom filter into the global index to reduce the communication cost. LocationSpark only keeps frequently accessed data in memory which reduces the chances of an overflow. However, it does not have support for SQL-like queries. SparkGIS [26] supports several dynamic partitioning algorithms (Fixed-Grid, Binary-Space, Quad-Tree, Strip-based, Hilbert-Curve, and STR), which mitigates the data distribution skew across the cluster. Like Simba and LocationSpark, it also incorporates multi-level (global and local) R*-tree indexing, which can be pre-generated or on-demand local in-memory indexing. Like LocationSpark, it also keeps data in-memory as much as possible to avoid running out of memory.

However, these five spatial data processing solutions are not fully compliant with the ISO standard and OGC specifications. Elcano [84] implements ISO and OGC-compliant 2D geometry data types, spatial functions, and operators based on Spark SQL [24]. It supports three indexation methods, GeoHash, R-Tree, and a combination of both (hybrid) [83]. However, the authors did not include any information regarding spatial data partitioning and query execution process of Elcano.

All these Spark-based systems discussed above are only for spatial data processing. STARK [114] integrates spatio-temporal support to Spark RDDs. It supports two spatial partitioning techniques (fixed grid, binary space) to distributed data across the cluster. Unlike other systems, it allows two modes of R-tree indexing, where the live index is built for each partition during query execution, and persistent indexing allows to create and save indexed RDD into disk or HDFS for future use. In addition to spatial join and kNN query, it supports DBSCAN clustering. STARK also extends Pig Latin for declarative spatio-temporal queries, called Piglet. DiStRDF [174] is a distributed system for processing spatio-temporal RDF data. It consists of two layers, where the storage layer enables fast retrieval and high availability of data by storing encoded RDF triples into HDFS and dictionary of mapping values into Redis [207]. The processing layer is based on the Spark query engine responsible for parsing, planning, and executing SPARQL queries, where Apache Jena [18] is used as a query parser. It uses spatio-temporal range partitioning to distribute 1D encoded RDF triples. It also supports Hilbert and Z-order hashing for indexing RDF triples. However, it only supports spatio-temporal point data.

As it is mentioned in Section 6.1, trajectory data is quite different from discrete spatio-temporal points. Therefore, the performance of processing trajectory data using STARK and DiStRDF will not be effective. Also, these systems work well for historical static data and require re-partitioning.
the whole dataset when the dataset has changed or updated. TrajSpark [282] always keeps the global index in the main memory and updates the global index when new data arrives using the time-decay model by partitioning only a batch of new data. TrajSpark also stores the updated global index into a disk to protect it from any future system failures. In TrajSpark, first, the raw trajectory points (RDD) are partitioned using a Quad-Tree/KD-Tree. Then, a local hash index is added to each partition, which creates IndexTRDDs. Finally, a multi-level hybrid global index (level1 - temporal, level2 - spatial, level3 - B+Tree) is built for each partition. However, it does not have any support for SQL-like queries. Zeyuan et al. [221] developed DITA, which supports both SQL and Dataframe API for trajectory analysis. DITA adopts the STR [146] partitioning strategy to create balanced partitions of trajectory points. Like TrajSpark, it also uses multi-level (global - R-Tree and local - Trie-like) indexing to expedite the query performance. Besides, DITA has developed a cost model to reduce inter-worker transmission costs and to balance the workload. Like DITA, UITraMan [75] also uses STR partitioning, but it has adopted R-Tree for both local and global indexing. Along with global indexing, UITraMan maintains a meta table to store information related to moving objects and partitions in order to improve query efficiency. Unlike other systems, UITraMan incorporates a data processing pipeline that includes data loading, preprocessing, extraction, and analysis. Since the performance of Spark-based systems is affected by GC (garbage collector) overhead due to the on-heap data caching of Spark, UITraMan has added an off-heap key-value store, Chronicle Map [228], into the block manager of Spark. Chronicle Map always keeps data in an off-heap cache, which reduces GC overhead of Spark and ensures data persistence on run-time.

Among TrajSpark, DITA, and UITraMan, only TrajSpark alleviates the overhead of re-partitioning the whole dataset when a new batch of dataset arrives. Thus, TrajSpark achieves near real-time trajectory processing capability, but it is not a system developed for processing real-time trajectory streams. Besides, this new batch of data is loaded as RDDs in Spark, which are immutable, and any updates on RDD create a new RDD, which is costly. Dragoon [93] is a hybrid system for processing both historical (offline) and streaming (online) trajectories. The offline module of Dragoon is similar to UITraMan, but Dragoon has utilized Chronicle Map in such a way that it works for both historical and streaming trajectories. In addition, a mutable RDD (mRDD) model is designed so that data can be updated later, which is key to the hybrid storage of Dragoon. In Dragoon, data partitioning (ID, spatial: Grid, STR, and temporal), indexing (two-level: R-Tree), and trajectory queries (ID, range, and kNN) are developed for both offline and online modules. Moreover, the data processing pipeline provides support for both historical and streaming trajectories.

Other than these systems, GeoMesa [152] has recently added support for Spark. However, all these systems are for processing vector spatial and spatio-temporal data. None of these systems has support for raster data except Apache Sedona. GeoTrellis [76] is a Scala library that enables Spark to process spatial raster data. It also has limited support for vector data. It can store into and query raster data from HDFS, S3, Accumulo, Cassandra, and HBase. Recently, Eldawy et al. presented a spatio-temporal data analytics system, Beast [79], that supports both vector and raster data with multidimensional data types and partition and index structures.

6.3 NoSQL-based Big Spatio-temporal Infrastructures

A number of big spatio-temporal data processing systems have been developed by utilizing NoSQL databases in the last couple of years, such as MD-HBase [175], Distributed SECONDO [170], GeoMesa [94, 124, 152], BBoxDB [172, 173], THBase [202], TrajMesa [150], and JUST [148]. The detailed feature matrix of these systems is presented in Table 5.

MD-HBase [175] extends HBase [17] to support spatio-temporal queries (range and kNN). It applies linearization (e.g., Z-Ordering) to transform multi-dimensional locations data (id, lat, lon, time) into 1D space for efficient indexing. A multi-dimensional index structure (Quad-tree,
KD-tree is layered on top of a key-value store, which allows real-time processing of range and kNN queries. MD-HBase achieves high insertion throughput, which is important for location-based applications. GeoMesa [152] is a spatio-temporal data processing system built on top of a distributed key-value store, Accumulo [94]. Like MD-HBase, GeoMesa also linearizes the keyspace by transforming multi-dimensional data (location, timestamp) into 1D keys using space-filling curves. It creates a spatio-temporal index using GeoHash and timestamps. At present, GeoMesa supports a set of indexing techniques, such as spatial (Z2 and XZ2), spatio-temporal (Z3, XZ3), ID index, and attribute index. Later, GeoMesa [124, 152] has added support for HBase, Google BigTable, Cassandra, Kafka, and Spark. Since traditional key-value stores with multi-dimensional support could be expensive to store and query non-point spatial data (e.g., polygons, lines), Jan et al. [172] have proposed a distributed and scalable key-bounding-box-value store called BBoxDB. Unlike traditional key-value stores, BBoxDB stores each value with an n-dimensional axis-parallel bounding box, which defines the location of the value in space. It uses space partitioning (Grid, KD-Tree, Quad-Tree) and multi-level indexing (global: KD-Tree, local: R-Tree) to store and organize the data across the cluster of nodes. However, BBoxDB only supports spatial-join queries, which can be executed locally on co-partitioned data.

Distributed SECONDO [170, 171] is a general-purpose DBMS, which can process relational, spatial, and spatio-temporal (including trajectory) data. It integrates the highly scalable and available key-value store Apache Cassandra and moving objects database SECONDO [112], where Cassandra is used as distributed data storage and the SECONDO as a query processing engine. Previously, SECONDO was integrated with Hadoop in Parallel SECONDO, but suffered due to centralized management. Besides, Parallel SECONDO [113] does not support high update rates. Distributed SECONDO achieves high update rates by integrating Cassandra. In addition, it supports both SQL-like and executable query. JUST (JD Urban Spatio-Temporal) [148] incorporates the power of HBase [17], GeoMesa [94, 124, 152], and Spark [276] into one system, where, HBase is used as an underlying storage structure, GeoMesa as an indexing tool, and Spark as a query execution engine. Along with indexing strategies of GeoMesa, JUST introduces two new indexing techniques, Z2T and XZ2T and efficient compression mechanism that improves the query performance significantly. An SQL-like query language, JustQL was also developed from scratch. Unlike in-memory systems, JUST only loads the necessary data into memory. Though other NoSQL-based systems (mentioned before) have achieved high update rates, their performance is hindered by disk latency. JUST has improved query efficiency by utilizing the main memory.

ACM Computing Surveys, Vol. 54, No. 10s, Article 219. Publication date: November 2022.
TrajMesa [149, 150] has adopted GeoMesa to develop a trajectory storage engine, where a horizontal storage schema (H-Store) is proposed for efficient trajectory data management. Instead of storing each point of a trajectory as a separate entry in a key-value store (V-Store), H-Store allows storing an entire trajectory in one-row with compression. Hence, in addition to reducing the storage size significantly, H-Store also improves the query efficiency by reducing disk I/O in TrajMesa. To perform a set of SQL-like queries (ID-Temporal, Range, kNN, and Similarity) on trajectory data efficiently, TrajMesa introduced ID temporal, XZT, and extended XZ2 indexing, XZ2+. Most importantly, TrajMesa incorporates a module for trajectory preprocessing [215] containing functions for noise filtering, segmentation, stay point detection, map matching, etc. As discussed before, if we split and store the trajectories of the same moving object (MO) into different partitions on different nodes of a cluster, the query processing efficiency of trajectory processing systems [75, 282] will be impacted negatively. To address this issue, THBase [202] has proposed a segment-based data model and a MO-based partitioning model for efficient trajectory storage management in HBase. THBase consists of three modules, T-table, L-index, and a query processing module. T-table is a container of trajectory data in which adopted the MO-based partitioning model, whereas, L-index is a local spatio-temporal index structure that consists of two levels (level1: time index, level2: Quad-tree based multi-level grid). Finally, the query processing module supports single-object, spatio-temporal range, and kNN query.

6.4 Python Libraries as Big Spatio-temporal Infrastructures

Python is one of the most popular data analytics platforms today. The PyData stack is rich in terms of supported libraries, but most of these libraries are developed to execute on a single CPU core and to process data that fits in main memory. Therefore, Python does not scale well for processing big data. One can utilize PySpark [73] for processing data on Spark. However, PySpark is added as an extra layer on top of Spark, and therefore, when Python code is executed using PySpark, the code is first compiled into Java code and then run on JVM. Thus, PySpark adds an extra overhead in computation. On the contrary, DASK [72] is a Python library for parallel and distributed computing that scales Python natively. DASK not only scales Python across distributed nodes of a cluster, but it also parallelizes tasks in a single node by utilizing multiple CPU cores.

Python also supports a rich set of libraries for processing spatial data. A detailed review of these libraries is presented in Section 7.2. However, these libraries are also slow and not scalable to process big spatial data. DASK does not have a native module for processing spatial data. One can utilize DASK DataFrame or low-level capabilities of DASK with existing spatial libraries to process spatial data. Previously, developers have tried to improve the performance of GeoPandas through Cython [38], which allows GeoPandas to access the GEOS library directly. However, Cythonizing only utilized one core of a node effectively [63]. Hence, the main challenge is to use multiple CPU cores or distributed nodes of a cluster. Recently, DASK-GeoPandas [226] was developed to parallelize GeoPandas with DASK, which organizes many GeoPandas dataframes like DASK-DataFrame. However, this is an experimental project and currently partitions dataframes by rows. Thus, spatial operations will not return the correct results in a distributed environment without using spatial partitioning. XArray [122] is a Python package for working with labelled multidimensional arrays, which are efficient for processing scientific datasets (e.g., netCDF, GeoTiff). Since XArray is tightly integrated with DASK for parallel computation, we can utilize XArray for processing spatial raster data. A few other projects, such as dask-geomodeling [64] and dask-rasterio [141], also provide support for spatial raster data. Moreover, dask-geomodeling has a module for spatial vector data.

As mentioned before, Hadoop is an efficient framework for processing big data, but hindered by the I/O bottleneck. In this context, as Spark keeps data always in-memory and does not require to write intermediate results back to disk, it became an efficient and popular data
processing framework. Currently, CPU-based in-memory systems like Spark suffer due to the bottleneck of processing complex workloads (e.g., deep learning on massive datasets), and this bottleneck is due to the CPU itself. Compared to CPUs with a few cores and lots of cache memory, GPUs are composed of hundreds of cores, high bandwidth memory (up to TB/s), and high-speed hardware interconnections (e.g., bidirectional GPU to GPU bandwidth up to 300 GB/s). Also, GPUs can scale up to 16x in a single node [139]. RAPIDS [241] is a collection of libraries and APIs that bring the power of GPUs for processing big data in Python. RAPIDS supports a wide range of libraries for Python developers, such as analytics (cuDF, cuIO), machine learning (cuML), graph analytics (cuGraph), deep learning (PyTorch, TensorFlow, MxNet), spatial analytics (cuSpatial), visualization (cuxFilter, pyViz, plotly), and other libraries. These libraries are literally replicated versions of the existing Python libraries, which opens doors to Python users to use the processing power of GPUs without knowing low-level CUDA implementations. Using RAPIDS, we can achieve vertical scalability easily. Since RAPIDS integrates DASK, horizontal scalability can be achieved on a single node as well as in distributed nodes of a cluster.

cuSpatial is a spatial module of RAPIDS, which is still in the early-stages of development, but growing rapidly. cuSpatial integrates with RAPIDS dataframe cuDF to use GPUs massive parallelism and high memory bandwidth for performing spatial operations. Developers can also use cuSpatial and cuGraph together for spatial and spatio-temporal analytics. cuSpatial is implementing spatial features in four layers that include geometry types, spatial operations, indexing, and querying. The current version of cuSpatial (v0.16) can model some basic geometry types such as points, polylines, polygons, and shape primitives. It supports Quad-Tree indexing for performing various spatial operations, such as point-in-polygon, Haversine distance (distance between points), and Hausdorff distance (distance between trajectories). Currently, it supports spatial window and nearest polyline queries. Since cuSpatial is working seamlessly with cuDF, users can use various data formats, such as CSV, Parquet, Shapefiles, JSON, and more.

In terms of support, Spark lacks data visualization and deep learning libraries. Whereas, both DASK and RAPIDS have support for data visualization and deep learning libraries (PyTorch, TensorFlow, Keras). At present, there is no native SQL query support in DASK, but Watson et al. has recently presented DaskDB [256], which integrates support for in-situ SQL queries. However, one can run SQL queries on both Spark (Spark SQL) and RAPIDS (blazingSQL). Though all of them have support for machine learning libraries, DASK (Dask-ML) and RAPIDS (cuML) libraries are more popular compared to Spark (MLib).

6.5 Other Big Spatio-temporal Infrastructures

A number of distributed spatial data systems were developed by extending existing infrastructures other than Hadoop, Spark, and NoSQL-databases. Sphinx [77, 78] extends the core of Impala [137], which is an SQL engine for Hadoop. Sphinx has adopted the ANSI-standard SQL interface and built spatial support in four layers of Impala. It introduced spatial data types, OGC-compliant spatial predicates and functions, and commands to create spatial indexes in the query parser layer. Two-level indexing (global: R-tree/R+-tree, local: R-tree) is added in the storage layer. Finally, Sphinx modifies the query planner and the executor layer to add support for spatial join and range queries. Since SQL-like queries of most big spatial data processing systems are not ANSI-standard and not efficient as spatial RDBMS, Sphinx achieves good performance over systems like SpatialHadoop. AsterixDB [6, 10, 11] incorporates LSM-based data storage and a set of indexing techniques including B+-tree and R-tree. It supports a complete query language, AQL, that uses Hyracks [45] as a query execution engine. A rich set of built-in data types, including spatial and temporal data, allows users to perform spatial, temporal, and spatio-temporal queries. Currently, it also supports
the SQL++ query language, which is very similar to SQL, but for semi-structured data (e.g., JSON). Like Spark, Apache Ignite [13] is also a distributed in-memory computing platform, but has limited spatial supports, see Table 6. Its spatial module supports geometry data types (point, line, and polygon), a limited form of querying on geometry data (intersection operation), and spatial indexing (R-tree). The main limitation of Ignite is that it does not support any spatial partitioning to distribute data across the clusters. Therefore, the result returned from Ignite for any spatial query is not accurate. Alam et al. [4, 5] introduced SpatialIgnite as extended spatial support for Ignite. They have added a spatial library containing all the OGC-compliant spatial predicates and analysis functions and introduced two spatial data partitioning techniques (fixed grid and Niharika [206]).

There are a few spatial database systems that have been developed from scratch, such as SciDB [37, 49, 62, 231], RasDaMan [33–35], and DISTIL [161, 190]. Both RasDaMan and SciDB are specialized database systems developed from scratch for scientific computing. These systems are implemented using a multi-dimensional array data model and are efficient for processing spatial raster data. Besides, these systems support SQL-like queries. The other array databases for processing large-scale spatial raster data are Google Earth Engine [104] and ChronosDB [213, 278], in which ChronosDB is the most recent one. We refer to [36, 117, 279] to learn more about array data databases for processing big spatial raster data.

On the other hand, DISTIL is implemented based on APGAS (Asynchronous Partitioned Global Address Space) programming model. Its efficient data partitioning and distributed multi-level spatio-temporal indexing expedites the performance of range and kNN queries. DISTIL achieves a high rate of updates by incorporating LSM-Tree [179] based key-value store LevelDB [233] as a local data store in each node of a cluster. Besides, the data in the local store is periodically synchronized with the distributed persistent global store, HDFS. At present, DISTIL does not have any support for SQL-like queries. Although data processing systems built from

---

1. APGAS - Asynchronous Partitioned Global Address Space.
2. RRR - Row-wise Round-Robin Partitioning.
3. MDR - Multi-dimensional Range Partitioning.
scratch for a specific purpose (e.g., spatial data processing) can achieve good performance, it is always challenging to use them as a general-purpose system [82]. Moreover, the code base of these systems is frequently immature and difficult to extend.

The big spatial data processing systems (except Dragoon [93]), which are developed based on Hadoop, Spark, or other platforms, can only store and process historical static spatial data. However, a wide range of location-based services require real-time processing of spatial data streams. Also, the indexing of these systems does not support high update rates to adjust with newly arriving streams. Besides, these services demand dynamic workload distribution. GeoFlink [220] extends Apache Flink to add support for processing spatial data streams. It has introduced grid-based dynamic indexing to perform continuous queries (range, kNN, and join). Tornado [157, 158], PS2Stream [58], and SSTD [55] have extended Apache Storm for processing spatio-textual (e.g., geo-tagged tweets) data streams. Tornado has added a two-level (global spatial and local spatio-textual) adaptive indexing layer for dynamically distributing data and query workloads. However, global indexing (A-Grid) and the cost model for load balancing of Tornado works well with continuous queries, not snapshot queries. PS2Stream has also proposed optimal workload partitioning and dynamic load balancing strategies for continuous subscription queries. Hence, SSTD has introduced QT-tree (a Quad-tree variant) global indexing and a set of local indexing for both continuous and snapshot queries (range, kNN, and top-k). STAR [56, 57] also supports snapshot and continuous aggregate queries (Algebraic, Tok-K) based on spatial, textual, and temporal constraints. Among them, Tornado and STAR support SQL-like queries. HASTE [265] is the most recent system for spatio-textual streaming data, which is deployed on top of Apache Flink. It has presented a global index and a workload partitioning method that efficiently utilizes both textual and spatial attributes compared to other spatio-textual streaming systems. Currently, it only supports continuous range queries.

At present, most major commercial data management systems have some form of spatial support. For example, Google BigQuery [60] has a GIS module (BigQuery GIS) to perform ANSI-standard SQL queries on large spatial datasets. Users can also visualize BigQuery results using BigQuery GeoViz and Google Earth Engine. Similarly, one can run standard SQL queries on Amazon Athena [12], Amazon Redshift [44] or Microsoft Azure for processing spatial data.

6.6 Future Research Directions

Considering the volume of spatial raster data generated from various sources (e.g., earth sensors, satellites) and the importance of this data in many application domains, more research is required for processing spatial raster data. There is a scope to improve the efficiency of SQL-like queries and the visualization capabilities of existing or new infrastructures. Also, there will be demand in the coming years to include support for visualizing big spatial data on web platforms. Recently, a few big data stream processing platforms (e.g., Flink, Storm) have extended to integrate support for processing spatial or spatio-textual data streams. Hence, there is a scope to explore spatio-temporal or trajectory data streams. Currently, there is a demand to incorporate SQL-like query engine and a dedicated spatial library in DASK. Since the spatial library (cuSpatial) of RAPIDS is in the early stage, there will be more research in the coming future to add more features for processing spatial and spatio-temporal data in GPU. A few performance benchmarks have been proposed to evaluate big spatial data systems, but there is a scope to work on benchmarking discrete spatio-temporal and trajectory data processing systems. Moreover, future big data infrastructures will include machine learning and deep learning models to process spatial data. For example, researchers have already started using machine learning for indexing spatial data [147, 189, 201]. Therefore, there is scope to develop learned indexes for spatio-temporal (or trajectory) data.
7 PROGRAMMING AND SOFTWARE TOOLS FOR SPATIO-TEMPORAL ANALYSIS

The infrastructure we use to store spatio-temporal datasets is one of the key aspects of any spatial application domain. However, the main goal is to extract value by analyzing these data. Analyzing data may sometimes simply involve using the available geospatial tools and writing some code for storing, querying, analyzing, and visualizing data. Other usage cases involve developing libraries or packages for specific purposes like spatial I/O or spatial regression. In each of these cases, one essential question comes into our mind: which programming language (or languages) to use to meet our purposes. This decision is frequently driven by our goals. For example, if we want to develop a system that requires heavy-weight development, we will look for a language that is fast and efficient like C/C++ or Java. However, if a developer wants to extend an existing system, the developer will most probably use the language on which the system was built. On the contrary, if our goals involve performing data processing, analysis and visualization, we need a language that provides a rich set libraries and packages good at these tasks, like Python or R. Moreover, many spatial systems (e.g., ArcGIS) leverage more than one language since some spatial features may be better supported by some languages than others.

The most popular libraries for modeling spatial data in use today in GIS applications and spatial data processing systems are developed either by Java or C/C++, such as JTS (Java) [153], GEOS (C++) [183], Google S2 (C++) [103], ESRI Geometry API (Java) [88], and Spatial4j (Java) [151]. Most of the big spatial data processing systems have also utilized these libraries to model spatial data, such as SpatialHadoop (JTS, ESRI Geometry API), GeoSpark (JTS), and GeoMesa (JTS, Spatial4j). Spatial RDBMSs like PostGIS or SpatialLite have used GEOS for modeling spatial data. Even libraries and packages of Python or R have also utilized the GEOS library for modeling spatial data. Recently, researchers have also proposed benchmarks [188, 281] for computational geometry libraries used in data processing systems for spatial data exploration.

At present, almost all popular programming languages have some support in terms of libraries or tools [142], which makes it easier to develop geospatial applications. However, C/C++ and Java are still top in the game for heavy-weight spatial system development. On the other hand, Python or R provide ease of programming and a rich set of libraries and packages for modeling, analyzing, and visualizing spatial data for a wide range of application domains. But if we need to work for the web, then we may need to get support from JavaScript, Python, and other web-related languages. It is similar for mobile development; we may need to choose language related to mobile operating systems, such as iOS or Android.

In this section, we perform a comprehensive review of widely used libraries and packages of Python and R for modeling, analyzing, and visualizing spatial, spatio-temporal, and trajectory data. This section will also discuss two popular software (ArcGIS and QGIS) for spatial data processing.

7.1 The R Ecosystem for Spatio-temporal Data Analysis

R is one of the most used languages in data science. From its inception, R is more focused on data analysis and statistical tasks, and therefore, it is more popular with academicians, statisticians, engineers, and scientists, who do not even have prior computer programming knowledge. In terms of the extension of libraries and packages for data analysis, R is richer than its counterparts. A rich set of packages and libraries are also available in R for analysis and visualization of spatial, temporal, spatio-temporal and trajectory data. In addition, R provides interfaces to spatial database systems, GIS software, and big data processing platforms. The R ecosystem for spatio-temporal data analysis is summarized in Table 7.

Data Processing Infrastructures: The packages related to spatio-temporal data processing can be categorized into three main groups, namely spatial, temporal, and spatio-temporal. The
Table 7. R Ecosystem for Spatio-temporal Data Analysis

| Category                        | Libraries/Packages/Tools/API's                                      |
|---------------------------------|---------------------------------------------------------------------|
| Data Processing                 | - spatial vector data: sp, sf                                        |
|                                 | - spatial raster data: raster, terra                                 |
|                                 | - spatio-temporal data: spacetime, trajectories, stars              |
|                                 | - time series: xts, zoo, its, ts                                     |
|                                 | - spatial network: tidygraph                                         |
|                                 | - OSGeo libraries: GDAL (rdal), GEOS (rgeos), PROJ.4(proj4)          |
| Data Manipulation               | - dplyr, tidyr, rmapshaper                                          |
| Data Modeling                   | - gstat, CAST, mlr/mlr3, performanceEstimation                      |
| Visualization                   | - ggmap: spatial visualization with ggplot2                         |
|                                 | - tmap: thematic maps in R (static, animated and interactive)       |
|                                 | - leaflet: JavaScript library for interactive web maps              |
|                                 | - mapview: interactive viewing of spatial data                      |
|                                 | - plotly: interactive web-based graphs via plotly.js                |
|                                 | - rasterVis: static visualization of raster data                    |
| APIs for GIS Software           | - RQGIS for QGIS, RPyGeo for ArcGIS                                 |
| and Spatial RDBMSs              | - RSAGA for SAGA, rgrass7 for GRASS                                 |
|                                 | - rpostgis for PostgreSQL/PostGIS                                   |
| APIs for                       | - Hadoop: Hadoop Streaming, RHadoop, RHIPE, ORCH                   |
| Big Data Platforms             | - Spark: SparkR, sparklyr                                           |
|                                 | - Apache Sedona (formerly GeoSpark): apache.sedona                  |

The **sp** [195] package consists of methods and classes to represent spatial data types and operations. However, spatial features developed in **sp** are not compliant with OGC simple features [177]. Moreover, as many features and functionalities of **sp** are directly dependent on OSGeo libraries [184] (such as GDAL, GEOS, and PROJ.4), if these libraries make any changes, it is difficult for **sp** to manage and maintain interfaces to these libraries due to a lack of simple features. The package **sf** (simple features) [192] provides classes and methods for spatial vector data, which supersedes the **sp** package. Its features are OGC-compliant and provide direct interfaces to the GDAL, GEOS, and PROJ.4 libraries. Therefore, if we use **sf**, we do not need to load these external libraries into R code. In addition, **sf** has many advantages over **sp**, that include faster I/O operations, improved visualization, compatible with the tidyverse collection of packages (e.g., **dplyr**), **sf** objects can be treated as data frames and finally, the functions in **sf** have a more consistent naming that makes it easier to use in the code [154]. Not surprisingly, **sf** is quickly being adopted as the backbone in many other packages related to spatial data analysis. The **raster** [121] package supports classes and a large set of functions to create, read, write, manipulate, and process raster data. This package can also process large raster datasets that are too big to fit in the main memory. The **terra** [120] package is a new package for processing raster data in R, containing similar functionality as the **raster** package. However, due to its simplicity and faster operation, **terra** will replace the **raster** package soon. The packages **stars** [193] or **spacetime** [191] are used for processing spatio-temporal data, but **spacetime** also has support for trajectory data. The package **trajectories** [194] was specifically developed for the analysis of trajectory data. These packages are dependent on packages for the analysis of time series (e.g., **xts** [216], **zoo** [280]) and spatial data (e.g., **sp**, **sf**).

**Data Manipulation:** The package **dplyr** [260] provides a grammar for data manipulation. It contains a set of functions (verbs) to manipulate data in dataframes, such as adding new columns, selecting specific columns, filtering and re-arranging rows, summarizing data, and other functions.
As the sf package is compatible with dplyr, the functions of dplyr can manipulate spatial objects in sf. In addition to manipulating in-memory dataframes, dplyr can also manipulate data stored in relational databases (using dbplyr [261]) or large datasets stored in Spark (using sparklyr [155]).

**Data Modeling:** The gstat [110] package is used for statistical modeling, prediction, and simulation of spatial and spatio-temporal data, which is also dependent on sp package. The CAST [163] package provides functions to improve spatio-temporal modeling tasks using caret package [140]. The mlr [41] package contains a wide range of machine learning algorithms for modeling data. However, due to its complex design, it is difficult for the developers to maintain and add new features in mlr. Besides, as some dependent packages of mlr have changed their features in the meantime, the developers of mlr could not manage to update the mlr accordingly. The mlr3 [143] is a successor of mlr, and it is a generic, object-oriented, and extensible framework that solves the above problems. The package performanceEstimation [249] also allows for predictive model development and tuning and moreover, contains specific routines for handling predictive tasks with time dependant data, like for instance sliding and growing window model building schemas.

**Data Visualization:** Mapping is one of the best ways to present the findings of spatial data analysis research. Though most of the spatial mapping packages were dependent on sp package before, a few of them (such as tmap [244], leaflet [59], and mapview [23]) are already supporting the classes of the sf package [154]. Spatial maps can be static or interactive and animated. The widely used static mapping tools are tmap and ggmap [132], but tmap is also used for interactive mapping. There is a wide range of packages for interactive and animated maps, such as leaflet, mapview and plotly [225]. Besides, rasterVis [198] is a package for static raster data visualization.

**APIs for GIS Software and Spatial RDBMSs:** R is very rich in terms of supported libraries and packages for analysis spatial data, but it is neither a spatial database system nor a powerful standalone GIS software tool. Besides, R packages are not capable of processing large scale spatial data. Therefore, the integration of R with GIS software and spatial RDBMSs extends the capabilities of R for processing spatial data. As a result, R users can use hundreds of algorithms from GIS software and can process data stored in database systems. RQGIS [167] establishes an interface between R and QGIS [200] by utilizing Python API for QGIS. As QGIS has already integrated other popular GIS software (such as SAGA, GRASS, and more), integrating R with QGIS brings the power of all these software into R using only one package, RQGIS. However, one can use dedicated APIs to access complete support of each of these GIS software that include rgdal [43] for GDAL, RSAGA [47] for SAGA, and rgrass7 [42] for GRASS. R users can also access functionalities of another popular commercial GIS software ArcGIS through the RPyGeo [48] package. Since the data analysis and visualization support of spatial RDBMS is limited, the integration of R with spatial RDBMSs is beneficial for both R and spatial RDBMSs users. rpostgis [50] package provides R with an interface to access a popular open-source database system, PostgreSQL/PostGIS.

**APIs for Big Data Platforms:** Traditional GIS software and spatial RDBMSs are not capable of handling today’s big spatial data. Therefore, R interfaces to these systems do not provide scalability and efficiency for processing this data. There are a number of R APIs that provide access to big data platforms, such as Hadoop and Spark. We can use Hadoop Streaming [20], RHadoop [212], RHIPE [71], and ORCH [181] API to run MapReduce jobs using Hadoop or directly accessing HDFS within the R programming environment. The sparklyr [155] and SparkR [252] packages are used as an interface between R and Spark. The package sparklyr is compatible with dplyr and allows R users to access the built-in machine learning algorithms of Spark. SparkR is a native Spark frontend for R users that provide access to all Spark libraries. Besides, these packages also allow R users to access HDFS directly. However, both Hadoop and Spark do not have native support for processing spatial data. On the other hand, most R packages for spatial analysis are

ACM Computing Surveys, Vol. 54, No. 10s, Article 219. Publication date: November 2022.
developed for a single node. Therefore, we can implement a custom R package for Hadoop and Spark-based spatial data processing systems (see Section 6) to achieve scalability. For example, the apache.sedona [21] package allows R users to use Apache Sedona (formerly GeoSpark) [22] for big spatial analysis.

7.2 Python Ecosystem for Spatio-temporal Data Analysis

Python is a full-fledged general purpose programming language. However, Python has become one of the most popular programming languages for data science in the last decade. It is widely used for modeling, analyzing, and visualizing data in both academia and industry. Researchers and organizations are continually working to develop new tools, libraries, and packages to process and analyze real-world data. Therefore, the community of Python is growing as well as the capabilities of the language. Due to the rise of spatial data, Python has been adopted for processing spatial, temporal, and spatio-temporal data in the last decade. The integration of Python as the main scripting language by popular GIS platforms like ArcGIS and QGIS has expedited this process. Python also supports interfaces to big data computing platforms, Hadoop and Spark. The Python ecosystem for spatio-temporal data analysis is summarized in Table 8.

The SciPy stack [243] consists of a set of libraries for scientific computing in Python. Specifically, the SciPy [243], Numpy [237], and Pandas [257] libraries have been used as core packages in data science. These packages are also essential for the analysis of spatio-temporal data since most of the libraries in the Python spatial stack depend on them.

Data I/O: The spatial input/output libraries of Python are developed using the existing C library, GDAL (Geospatial Data Abstraction Library) [99], which supports a wide range of raster and vector data formats. Therefore, Python spatial I/O libraries also support these data formats. Fiona [102] interfaces to the OGR (OpenGIS Reference Implementation) layer of GDAL for reading and writing spatial vector data of various formats, such as Shapefile, GeoJSON, etc. The library rasterio [240] interfaces to GDAL for raster functionality. It relies on Numpy for efficient processing of raster formats, such as GeoTIFF, netCDF, and other formats.

Data Processing: A set of libraries is available in Python for processing spatial and spatio-temporal data. The library Shapely [222] provides functions for manipulation and analysis of vector geometric objects, and is based on the widely used GEOS [183] library. GeoPandas [100] is a spatial extension of Pandas. It uses the Shapely, Fiona, and pyproj [258] libraries to add spatial support in the popular data analysis and manipulation tool, Pandas. SciPy.Spatial also provides algorithms and data structures for spatial analysis. Whereas, rasterstats [199] contains functions for zonal statistics and interpolated point queries for summarizing spatial raster datasets using vector geometries. Moreover, one can use pyspatial [245] for both raster and vector data, sptemp [32] for spatio-temporal vector data, and MovingPandas [105] and traja [224] for trajectory data analysis.

Statistical Analysis and Modeling: PySAL [211] is an open-source spatial analysis library with a primary focus on vector data. The functionality of PySAL covers a wide range of areas, such as methods to detect spatial clusters, hot spots and outliers, spatial regression, statistical modeling, spatial econometrics, space-time analysis, visualization, and more. The current version of PySAL (V2.X) consists of four domains, which include PySAL core (pysal.lib), exploratory spatial data analysis (pysal.explore), spatial statistical models (pysal.model), and geovisualization (pysal.viz). pysal.lib is the core library which contains data structures and algorithms for spatial I/O, spatial weights, computational geometry, and more. The pysal.explore library consists of modules for exploratory analysis of spatial and spatio-temporal data. The pysal.model is designed to model spatial relationships in data using different types of linear, generalized-linear, generalized-additive, nonlinear, multi-level, and local regression models. Finally, the pysal.viz layer supports
Table 8. Python Ecosystem for Spatio-temporal Data Analysis

| Category                  | Libraries/Packages/Tools/API’s                                                                 |
|---------------------------|-------------------------------------------------------------------------------------------------|
| **Core Libraries**        | - SciPy: core library for scientific computation                                               |
|                           | - NumPy: fundamental library for numerical computation                                          |
|                           | - Pandas: data structure and analysis library                                                   |
| **Data I/O**              | - GDAL: raster and vector I/O (interface to GDAL/OGR)                                           |
|                           | - Fiona: vector I/O (interface to OGR)                                                          |
|                           | - Rasterio: raster I/O (interface to GDAL)                                                       |
| **Data Processing**       | - GeoPandas: spatial extension of pandas                                                        |
|                           | - Shapely: spatial analysis of geometric objects                                                |
|                           | - scipy.spatial: spatial algorithms and data structures                                          |
|                           | - pyspatial: analysis vector/raster data                                                         |
|                           | - sptemp: spatio-temporal vector data analysis                                                   |
|                           | - Rtree: spatial indexing                                                                      |
|                           | - rasterstats: summarizing spatial raster datasets                                               |
|                           | - MovingPandas, traja: trajectory data analysis                                                   |
|                           | - pyproj: coordinate transformations (interface to PROJ4)                                       |
| **Statistical Analysis**  | - PySAL: spatial and spatio-temporal data analysis library                                      |
|                           | - statsmodels: statistical modeling                                                             |
|                           | - scikit-learn: machine-learning algorithms                                                     |
|                           | - scikit-image: algorithms for image (satellite) processing                                     |
|                           | - River: machine learning for streaming data                                                     |
| **Visualization**         | - Matplotlib: static and interactive visualization                                              |
|                           | - Seaborn: statistical data visualization                                                        |
|                           | - Bokeh: interactive visualizations for the web                                                 |
|                           | - Plotly: interactive visualizations for the web                                                 |
|                           | - Folium: visualizations via interactive leaflet map (leaflet.j)                                |
|                           | - Cartopy: visualize data on maps                                                                |
|                           | - ggplot: visualizations based on R ggplot2                                                      |
| **GIS Software**          | - ArcPy, ArcGIS API: python interface to ArcGIS                                                  |
|                           | - PyQGIS: Python interface to QGIS                                                               |
| **Big Data Platforms**    | - Hadoop: Hadoop Streaming, mrjob, Pydoop, Luigi, PyArrow                                        |
|                           | - Spark: PySpark                                                                                |

functionality to visualize spatially analyzed data (e.g., detected clusters or hot-spots). Besides, PySAL provides a toolkit for ArcGIS and a plugin for QGIS which allows using the functionalities of PySAL within these GIS software. Some desktop applications like CAST (Crime Analytics in Space-Time) and GeoDaSpace has also used a subset of PySAL. Moreover, PySAL is now available as a featured package in the distribution of Anaconda Python and Enthought Canopy [210]. statemodels [218] is another useful library for statistics, financial econometrics, or econometrics. It also supports models for time-series analysis. scikit-learn [196] is a library of a vast collection of supervised and unsupervised machine learning algorithms for clustering, classifications, regression, dimensionality reductions, and many more. It also supports functions for data loading, manipulation, and prepossessing. However, scikit-learn is a library for learning from historical data. River [165] is a machine learning library for streaming data. It combines the strengths of two previous libraries, Creme [116] and scikit-multiflow [166], under a revamped architecture to unify continuous development efforts. On the other hand, scikit-image [250] includes a wide
range of algorithms for image analysis, such as segmentation, transformations, restoration, metrics, feature selection, color space manipulation, filtering, morphology, and other algorithms.

**Data Visualization:** Like R, Python also supports a rich set of libraries for spatial data visualization and mapping. Matplotlib [126] is a key visualization library in Python, which can be used for creating static, interactive, and animated visualizations. The package Cartopy [162] uses PROJ.4, Shapely, and NumPy to provide a spatial mapping library on top of Matplotlib. Seaborn [219] is another library built on top of Matplotlib for statistical data visualization. A number of Python packages are also built to create interactive maps for the web, such as Folium [232], Bokeh [74], and Plotly [225]. Folium is a wrapper for the leaflet.js library for plotting interactive web maps. It includes a raster and a vector layer for visualizing through an interactive leaflet map. Similarly, Bokeh and Plotly are also developed as interactive visualization libraries for web browsers.

**Python for GIS Software:** The integration of Python as a main scripting language of ArcGIS [89] and QGIS [200] allows users to use the combined power of GIS software and Python for processing spatial data. The ArcPy package ships with a desktop version of ArcGIS and allows Python to access GIS tools with extensions, useful functions, classes, and modules for processing geospatial data. This package helps users to write Python scripts that can be run within ArcGIS or as standalone scripts. ArcGIS API for Python (called Pythonic GIS API) gives access to a wide range of modules, classes, and functions provided by ArcGIS Online and ArcGIS Enterprise. Similarly, we can use Python with QGIS through PyQGIS API. Other popular GIS software like GRASS GIS [106] and SAGA GIS [242] also supports APIs for Python.

**APIs for Big Data Platforms:** As it was mentioned in Section 1, spatial libraries and packages of Python were developed for processing data in a single-node computing environment. Therefore, we need to use parallel and distributed computing platforms like Hadoop and Spark for processing big datasets. There are a number of Hadoop APIs that allow Python users to access the Hadoop MapReduce paradigm and distributed file system, HDFS, which include Hadoop Streaming [214], mrjob (Yelp) [267], Pydoop [145], and Luigi (Spotify) [246]. One can write and run MapReduce jobs on Hadoop using all these APIs, but the Hadoop Streaming API ships with Hadoop as a native API. mrjob can also run MapReduce jobs locally without Hadoop for testing purposes. Pydoop is tightly integrated with Hadoop and provides full access to Hadoop APIs and direct access to HDFS via its HDFS API. Moreover, PyArrow also includes a client to access HDFS. On the other hand, PySpark [73] is a native Spark API that enables Python users to interact with the Spark programming paradigm for processing large datasets. Along with a rich set of Python libraries, this API also allows Python users to use built-in Spark libraries, such as MLlib, Spark Streaming, and Spark SQL and Dataframes. Besides, PySpark can process data stored in a distributed storage like HDFS. However, these Python APIs add extra overhead in computation during data processing as they are developed as a layer on top of Hadoop and Spark. Hence, DASK [72] and RAPIDS [241] have emerged as parallel and distributed Python libraries to mitigate this issue (see Section 6.4).

### 7.3 GIS Software

A GIS (Geographical Information System) is an integrated environment to capture, store, analyze, and visualize geographical data. GIS software is a useful tool for researchers, scientists, or practitioners, who want to extract inherent knowledge, patterns, and relationships from geographical data and analyze the data to address real-world problems. There are a few commercial and open-source GIS software available, such as ArcGIS [89], QGIS [200], GRASS [106], and SAGA [242]. ArcGIS and QGIS are the most popular GIS software among them.

ArcGIS [89] is a leading commercial GIS software developed by ESRI [90]. Whereas, QGIS (Quantum GIS) [200] is a popular open-source GIS software that supports similar functionalities to ArcGIS. Since QGIS has integrated some popular GIS software like GRASS [106], SAGA [242], and
OTB (Orfeo Toolbox) [109], one can use a subset of algorithms from these GIS software within QGIS. However, if we want to use full functionality of these GIS software systems, we need to stick with GRASS and SAGA [154]. Both ArcGIS and QGIS have support for a Python console that allows users to execute the functionality of GIS tools and Python within ArcGIS and QGIS. Also, both of them have an interface to Python and R. Therefore, we can use the functionality of ArcGIS and QGIS within Python and R programming environments. According to an experiment conducted by Debicka et al. [69] based on the buffer, convex hull, and intersection geometric operations, QGIS is faster than ArcGIS for processing spatial data. However, in terms of spatial mapping capabilities, ArcGIS is way better than QGIS. ArcGIS is also richer in terms of tools and supported algorithms, but one can extend the capabilities of QGIS by adding third-party plugins.

Like spatial RDBMSs, GIS software is also going through many changes over the years to adapt to this era of big spatial data. ESRI has released open-source GIS tools for Hadoop [87] to perform analysis on big spatial data by utilizing the distributed processing capability of Hadoop. ESRI has also introduced a Spark-powered GeoAnalytics toolbox for both ArcGIS server and desktop versions. GeoAnalytics Desktop brings parallel processing of data across multiple cores of a personal computer through ArcGIS Pro. Whereas, GeoAnalytics Server provides distributed processing of big spatial data across multiple nodes of a cluster running ArcGIS Enterprise. As Spark ships with ArcGIS, users do not need to install Spark separately. Besides, ESRI ArcSDE is an RDBMSs gateway that allows ArcGIS users to store, manage, and use spatial data in some popular databases, such as IBM DB2 and Informix, Oracle, Microsoft SQL Server, and PostgreSQL. In summary, GIS software is continuously adapting to process big spatial data.

7.4 Future Research Directions

Since libraries and packages of Python and R are developed for processing data in a single node computer system, these libraries are not suitable for processing big data. Python and R users could utilize the available APIs for big data platforms, such as Hadoop and Spark. However, we do not know how these libraries will perform with big data platforms since there is no comprehensive evaluation yet. We can also implement new APIs to use big spatial systems (see Section 6) that have been developed based on big data platforms. Currently, DASK and RAPIDS are promising platforms to process big spatial data for Python users. R users either need to use existing big spatial data processing systems or need to develop a system like DASK or RAPIDS. On the other hand, GIS software will be adding new features and modules to create analysis and mapping facilities for a wide range of new application domains. Since machine learning (ML) and deep learning (DL) algorithms and techniques are important for solving complex spatial problems, there will be more ML and DL models in GIS software in the future.

8 CONCLUSION

Due to the rise of spatio-temporal data volume and the significance of extracted knowledge in a wide range of application domains, plenty of research and development works have been done in the area of spatio-temporal data analytics in the last decade. Survey work is always pivotal for researchers to know and advance the state-of-the-art. In this survey, we have conducted a comprehensive study on the whole ecosystem of spatio-temporal data analytics, which covers spatial databases (SQL and NoSQL), big spatial data infrastructures, programming languages, and software tools. This study also addressed the importance, current demand, and future of spatio-temporal data analytics.

We argue that the research community needs to address a few areas of spatio-temporal data analytics in future research that include (i) integrating more support to model and analysis of spatial raster data, (ii) integrating more support for processing spatio-temporal (or trajectory) data
A Survey on Spatio-temporal Data Analytics Systems 219:29

streams, (iii) integrating or improving SQL-like queries, and (iv) adding more support for analysis and visualization in big spatio-temporal infrastructures. It is already evident that there will be more research on integrating AI, machine learning, and deep learning models in future big spatio-temporal infrastructures for uncovering hidden knowledge. There will be demand for integrating visualization support for big spatial data in web platforms. Besides, we think future infrastructures will be more application-specific, such as IoT, neuroscience, emergency management, transportation, and other applications. The usage of GPU in RAPIDS has shown significant speed-up in computation compared to other infrastructures, and therefore, more research is required in terms of using GPUs for spatio-temporal data analytics. We hope that the accumulated information in this study will be useful for researchers, practitioners, and developers who are currently working or who want to work in the area of spatio-temporal data analytics.

REFERENCES
[1] Rifaat Abdalla. 2016. Evaluation of spatial analysis application for urban emergency management. SpringerPlus 5 (12 2016).
[2] David W. Adler. 2001. DB2 spatial extender - Spatial data within the RDBMS. In VLDB. Morgan Kaufmann Publishers Inc., San Francisco, CA, 687–690.
[3] Ablimit Aji, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, and Joel Saltz. 2013. Hadoop-GIS: A high performance spatial data warehousing system over MapReduce. VLDB 6, 11 (2013), 1009–1020.
[4] Md Mahbub Alam. 2018. Parallel and In-Memory Big Spatial Data Processing Systems and Benchmarking. Master’s thesis. Fredericton, NB, Canada.
[5] Md Mahbub Alam, Suprio Ray, and Virendra C. Bhavsar. 2018. A performance study of big spatial data systems. In BigSpatial 2018. ACM, New York, NY, USA, 1–9.
[6] Abdullah A. Alamoudi, Raman Grover, Michael J. Carey, and Vinayak Borkar. 2015. External data access and indexing in AsterixDB. In CIKM’15. ACM, New York, NY, USA, 3–12.
[7] Louai Alarabi. 2019. Summit: A scalable system for massive trajectory data management. SIGSPATIAL Special 10, 3 (Jan. 2019), 2–3.
[8] Louai Alarabi, Mohamed F. Mokbel, and Mashaal Musleh. 2018. ST-Hadoop: A MapReduce framework for spatio-temporal data. Geoinformatica 22, 4 (Oct 2018), 785–813.
[9] Harriet Alexander and Nick Moir. 2019. The monster: A short history of Australia’s biggest forest fire. The Sydney Morning Herald (2019). https://www.smh.com.au/national/nsw/the-monster-a-short-history-of-australia-s-biggest-forest-fire-20191218-p53l4y.html. December 20, 2020.
[10] Sattam Alsubaiee, Yasser Altwowim, Hotham Altwajiry, Alexander Behm, Vinayak Borkar, Yingyi Bu, Michael Carey, Inci Cetindil, Madhusudan Cheelangi, Khurram Farazaa, Eugenia Gabriëlova, Raman Grover, Zachary Heilbronn, Young-Seok Kim, Chen Li, Guangqiang Li, Ji Mahn Olk, Nicola Onouse, Pouria Pirzadeh, Vassilis Tsotras, Rares Vernica, Jian Wen, and Till Westmann. 2014. AsterixDB: A scalable, open source BDMS. VLDB Endow. 7, 14 (Oct 2014), 1905–1916.
[11] Sattam Alsubaiee, Alexander Behm, Vinayak Borkar, Zachary Heilbronn, Young-Seok Kim, Michael J. Carey, Markus Dreseler, and Chen Li. 2014. Storage management in AsterixDB. Proc. VLDB Endow. 7, 10 (June 2014), 841–852.
[12] Amazon. 2020. Amazon Athena: A Serverless, Interactive Query Service. https://aws.amazon.com/athena/.
[13] Apache 2015. Apache Ignite: A Distributed In-Memory Computing Platform. https://ignite.apache.org/.
[14] Apache. 2016. Cassandra: Manage massive amounts of data, fast, without losing sleep. https://cassandra.apache.org/.
[15] Apache. 2019. Apache Storm: A Distributed Real-time Data Stream Processing Platform. https://storm.apache.org/.
[16] Apache. 2019. Apache Hadoop: An open-source distributed processing framework. https://hadoop.apache.org/.
[17] Apache. 2020. Apache Jena: A Semantic Web Framework for Java. https://jena.apache.org/.
[18] Apache. 2020. CouchDB: An Open Source NoSQL Database. https://couchdb.apache.org/.
[19] Apache. 2020. Hadoop Streaming API. https://hadoop.apache.org/docs/r1.2.1/streaming.html.
[20] Apache Software Foundation. 2021. Apache Sedona: A Cluster Computing System for Processing Large-scale Spatial Data. https://sedona.apache.org/tutorial/core-python/.
[21] Tim Appelhans, Florian Detsch, Christoph Reudenbach, and Stefan Woellauer. 2020. mapview: Interactive Viewing of Spatial Data in R. https://CRAN.R-project.org/package=mapview.
[24] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kahan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. 2015. Spark SQL: Relational data processing in Spark. In SIGMOD'15. ACM, New York, NY, USA, 1383–1394.

[25] Gowtham Athuri, Anuj Karpatne, and Vipin Kumar. 2018. Spatio-temporal data mining: A survey of problems and methods. ACM Comput. Surv. 51, 4, Article 83 (Aug. 2018), 41 pages.

[26] Furqan Baig, Hoang Vo, Tahsin Kuc, Joel Saltz, and Fusheng Wang. 2017. SparkGIS: Resource aware efficient spatio-temporal data processing extension. In SIGSPATIAL. ACM, New York, NY, USA, Article 28, 10 pages.

[27] Mohamed Bakli, Mahmoud Sakr, and Tayisr Hassan A. Soliman. 2019. HadoopTrajectory: A Hadoop spatiotemporal data processing extension. Journal of Geographical Systems 21, 2 (June 2019), 211–235.

[28] Mohamed Bakli, Mahmoud Sakr, and Esteban Zimanyi. 2019. Distributed moving object data management in MobilityDB. In SIGSPATIAL Workshop (BigSpatial’19). ACM, New York, NY, USA, Article 1, 10 pages.

[29] Jie Bao, Ruiyuan Li, Xiwen Yi, and Yu Zheng. 2016. Managing massive trajectories on the cloud (SIGSPACIAL ’16). ACM, New York, NY, USA, Article 10, 10 pages.

[30] E. Baralis, A. Dalla Valle, P. Garza, C. Rossi, and F. Scullino. 2017. SQL versus NoSQL databases for geospatial applications. In IEEE International Conference on Big Data. 3388–3397.

[31] Dominik Bartoszewski, Adam Piorkowski, and Michal Lupa. 2019. The comparison of processing efficiency of spatial data for PostGIS and MongoDB databases. In Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis. Springer International Publishing, 291–302.

[32] Daniel Baumann. 2019. sptemp: Python package for spatio-temporal vector data processing and analysis. https://github.com/BaumannDaniel/sptemp. Accessed on December, 2020.

[33] Peter Baumann, Andreas Dehmel, Paula Fritzado, Roland Ritsch, and Norbert Widmann. 1999. Spatio-temporal retrieval with RaDaMan. In VLDB. Morgan Kaufmann Publishers Inc., San Francisco, CA. 746–749.

[34] Peter Baumann, Paula Fritzado, Roland Ritsch, and Norbert Widmann. 1997. The RaDaMan approach to multidimensional database management. In ACM Symposium on Applied Computing (SAC’97). ACM, New York, NY, USA, 166–173.

[35] Peter Baumann, Dimitar Misev, Vlad Merticariu, and Bang Pham Huu. 2021. Array databases: Concepts, standards, implementations. Journal of Big Data 8 (2021), 1–61.

[36] J. Becla, D. Zhang, M. Stonebraker, and P. Brown. 2013. SciDB: A database management system for applications with complex analytics. Computing in Science & Engineering 15, 3 (May 2013), 54–62.

[37] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. 2011. Cython: The best of both worlds. Computing in Science & Engineering 13, 2 (2011), 31–39.

[38] V. Bhanumurthy, K. Ram Mohan Rao, G. Jai Shankar, and P. V. Nagamani. 2017. Spatial data integration for disaster/emergency management: An Indian experience. Spatial Information Research 25 (2017), 303–314.

[39] V. Bhanumurthy, G. Jai Shankar, K. Ram Mohan Rao, and P. V. Nagamani. 2015. Defining a framework for integration of geospatial technologies for emergency management. Geocarto International 30, 9 (2015), 963–983.

[40] Bernd Bischl, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. mlr: Machine learning in R. Journal of Machine Learning Research 17, 170 (2016), 1–5.

[41] Roger Bivand. 2019. rgrass7: Interface Between GRASS 7 Geographical Information System and R. https://CRAN.R-project.org/package=rggrass7.

[42] Roger Bivand, Tim Keitt, and Barry Rowlingson. 2020. rgdal: Bindings for the 'Geospatial' Data Abstraction Library. https://CRAN.R-project.org/package=rgdal.

[43] Nemanja Borić, Hinmenk Gildhoff, Menelaos Karavelas, Ippokratis Pandis, and Ioanna Tsalouchidou. 2020. Unified spatial analytics from heterogeneous sources with Amazon redshift. In SIGMOD. ACM, 2761–2784.

[44] V. Borkar, M. Carey, R. Grover, Nicola Onose, and R. Vernica. 2011. Hyracks: A flexible and extensible foundation for data-intensive computing. ICDE (2011), 1151–1162.

[45] M. B. Brahim, W. Drira, F. Filali, and N. Hamdi. 2016. Spatial data extension for Cassandra NoSQL database. Journal of Big Data 3 (2016), 1–16.

[46] Alexander Brenning, Donovan Bangs, and Marc Becker. 2018. RSAGA: SAGA Geoprocessing and Terrain Analysis. https://CRAN.R-project.org/package=RSAGA.

[47] Alexander Brenning, Fabian Polakowski, and Marc Becker. 2018. RPyGeo: ArcGIS Geoprocessing via Python. https://CRAN.R-project.org/package=RPyGeo.

[48] Paul G. Brown. 2010. Overview of SciDB: Large scale array storage, processing and analysis. In SIGMOD. ACM, New York, NY, USA, 963–968.

[49] David Bucklin and Mathieu Basille. 2018. rpostgis: Linking R with a PostGIS spatial database. The R Journal 10, 1 (2018), 251–268. https://journal.r-project.org/archive/2018/RJ-2018-025/index.html.
A Survey on Spatio-temporal Data Analytics Systems

[51] Jesinta Burton. 2020. ‘It was a line of fire coming at us’: South west firefighters return home. *Busselton-Dunsborough Mail* (2020). https://www.busseltonmail.com.au/story/6620315/it-was-a-line-of-fire-coming-at-us-firefighters-return-home/. February 7, 2020.

[52] Elasticsearch B. V. 2020. Elasticsearch: The heart of the free and open Elastic Stack. https://www.elastic.co/elasticsearch/. December, 2020.

[53] CARTO. 2020. CARTO: Unlock the power of spatial analysis. https://carto.com/. Accessed on October, 2020.

[54] João Pedro de Carvalho Castro, Anderson C. Carmiel, and Cristina Dutra de Aguiar Ciferri. 2018. A user-centric view of distributed spatial data management systems. In *GEOINFO*. MCTI/INPE.

[55] Yue Chen, Zhida Chen, Gao Cong, Ahmed R. Mahmood, and Walid G. Aref. 2020. SSTD: A distributed system on streaming spatio-textual data. *Proc. VLDB Endow.* 13, 12 (July 2020), 2284–2296.

[56] Zhida Chen, Gao Cong, and Walid G. Aref. 2020. STAR: A distributed stream warehouse system for spatial data. In *SIGMOD (SIGMOD’20)*. ACM, New York, NY, USA, 2761–2764.

[57] Zhida Chen, Gao Cong, and Walid G. Aref. 2021. STAR: A Cache-Based Distributed Warehouse System for Spatial Data Streams. ACM, New York, NY, USA, 606–615.

[58] Zhida Chen, Gao Cong, Zhenjie Zhang, Tom Z. J. Fuz, and Lisi Chen. 2017. Distributed publish/subscribe query processing on the spatio-textual data stream. In *ICDE*. 1095–1106.

[59] Joe Cheng, Bhaskar Karambelkar, and Yihui Xie. 2019. *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet'* Library. https://CRAN.R-project.org/package=leaflet.

[60] Google Cloud. 2020. Google BigQuery: Serverless, Highly Scalable, and Cost-effective Multi-cloud Data Warehouse. https://cloud.google.com/bigquery.

[61] Couchbase. 2011. GeoCouch: A Spatial Index for CouchDB. https://github.com/couchbase/geocouch.

[62] P. Cudre-Mauroux, H. Kimura, K.-T. Lim, J. Rogers, R. Simakov, E. Sorouch, P. Velikhov, D. L. Wang, M. Balazinska, J. Becla, D. DeWitt, B. Heath, D. Maier, S. Madden, J. Patel, M. Stonebraker, and S. Zdonik. 2009. A demonstration of SciDB: A science-oriented DBMS. *VLDB Endow.* 2, 2 (Aug 2009), 1534–1537.

[63] DASK Blog 2017. Fast GeoSpatial Analysis in Python. https://blog.dask.org/2017/09/21/accelerating-geopandas-1.

[64] dask-geomodeling 2019. On-the-fly operations on geographical maps. https://dask-geomodeling.readthedocs.io/.

[65] Citus Data. 2011. A distributed PostgreSQL as an extension for multi-tenant and real-time analytics workloads. https://github.com/citusdata/citus.

[66] Ali Davoudian, Liu Chen, and Mengchi Liu. 2018. A survey on NoSQL stores. *ACM Comput. Surv.* 51, 2, Article 40 (April 2018), 43 pages.

[67] Damião de Almeida, Cláudio Baptista, Fabio Andrade, and Amilcar Soares. 2020. A survey on big data for trajectory analytics. *ISPRS Int. J. of Geo-Information* 9 (2020), 88.

[68] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *Commun. ACM* 51, 1 (Jan 2008), 107–113.

[69] Justyna Debicka and Stanislaw Szombara. 2018. Comparative analysis of ArcGISs and QGISs in terms of the transformations’ runtime. *Geoinformatica Polonica* 17 (1 2018), 99–108.

[70] Jorge A. Delgado, Nicholas M. Short, Daniel P. Roberts, and Bruce Vandenberg. 2019. Big data analysis for sustainable agriculture on a geospatial cloud framework. *Frontiers in Sustainable Food Systems* 3 (2019), 54.

[71] DeltaRho. 2020. RHIPE: R and Hadoop Integrated Programming Environment. https://github.com/delta-rho/RHIPE.

[72] Development Team. 2016. Dask: Library for Dynamic Task Scheduling. https://dask.org.

[73] Development Team. 2016. PySpark: Spark Python API. https://spark.apache.org/docs/latest/api/python/.

[74] Development Team. 2020. *Bokeh: Python Library for Interactive Visualization*. https://bokeh.org/.

[75] Xin Ding, Lu Chen, Yunjun Gao, Christian S. Jensen, and Hujun Bao. 2018. UlTraMan: A unified platform for big trajectory data management and analytics. 11, 7 (March 2018), 787–799.

[76] Eclipse Foundation, Azavea, and contributors. 2016. GeoTrellis: A Scala Library and Framework for Spatial Raster and Vector Data. https://github.com/locationtech/geotrellis.

[77] Ahmed Eldawy, Mostafa Elganaeny, Ammar Bakeer, Ahmed Abdelmotaleb, and Mohamed Mokbel. 2015. Sphinx: Distributed execution of interactive SQL queries on big spatial data (*SIGSPATIAL’15*). ACM, New York, NY, USA, Article 78, 4 pages.

[78] Ahmed Eldawy, Mostafa Elganaeny, Ammar Bakeer, Ahmed Abdelmotaleb, and Mohamed Mokbel. 2017. Sphinx: Empowering impala for efficient execution of SQL queries on big spatial data. In *Advances in Spatial and Temporal Databases*. Springer International Publishing, 65–83.

[79] Ahmed Eldawy, Vagelis Hristidis, Saheli Ghosh, Majid Saeedan, Akil Sevim, A. B. Siddique, Samriddhi Singla, Ganesh Sivaram, Tin Vu, and Yaming Zhang. 2021. *Beast: Scalable Exploratory Analytics on Spatio-Temporal Data*. ACM, New York, NY, USA, 3796–3807.

[80] Ahmed Eldawy and Mohamed F. Mokbel. 2014. Pigeon: A spatial MapReduce language. *ICDE* (2014), 1242–1245.
[81] Ahmed Eldawy and Mohamed F. Mokbel. 2015. SpatialHadoop: A MapReduce framework for spatial data. In ICDE. IEEE Computer Society, 1352–1363.
[82] Ahmed Eldawy and Mohamed F. Mokbel. 2016. The era of big spatial data: A survey. 6, 3–4 (Dec 2016), 163–273.
[83] Jonathan Engélins, Thierry Badard, and Éveline Bernier. 2019. Enabling standard geospatial capabilities in Spark for the efficient processing of geospatial big data. In Geographical Information Systems Theory, Applications and Management. Springer, Cham, 133–148.
[84] Jonathan Engélins and Thierry Badard. 2018. Elcano: A geospatial big data processing system based on SparkSQL. In GISTAM. 119–128.
[85] Martin Erwig, Ralf Hartmut Güting, Markus Schneider, and Michalis Vazirgiannis. 1999. Spatio-temporal data types: An approach to modeling and querying moving objects in databases. Geoinformatika 3 (1999), 269–296.
[86] ESRI. 2008. GIS for sustainable agriculture. GIS Best Practices (2008).
[87] ESRI. 2013. GIS Tools for Hadoop: Big Data Spatial Analytics for the Hadoop Framework. https://esri.github.io/gis-tools-for-hadoop/. Accessed on October, 2020.
[88] ESRI. 2014. ESR Geometry API. https://github.com/Esri/geometry-api-java. Accessed on October, 2020.
[89] ESRI. 2020. ArcGIS: A Geographic Information System. https://www.arcgis.com/index.html.
[90] ESRI 2020. ESRI: Geographic Information Company. https://www.esri.com/. Accessed on December, 2020.
[91] Michael R. Evans, Dev Oliver, KwangSoo Yang, Xun Zhou, Reem Y. Ali, and Shashi Shekhar. 2019. Enabling Spatial Big Data via CyberGIS: Challenges and Opportunities. Springer Netherlands, Dordrecht, 143–170.
[92] Yi Fang, Marc Friedman, Giri Nair, Michael Rys, and Ana-Elisa Schmid. 2008. Spatial indexing in Microsoft SQL server 2008 (SIGMOD’08). ACM, New York, NY, USA, 1207–1216.
[93] Ziquan Fang, Lu Chen, Yunjun Gao, Lu Pan, and Christian Jensen. 2021. Drogon: A hybrid and efficient big trajectory management system for offline and online analytics. The VLDB Journal 2 (2021).
[94] A. Fox, C. Eichelberger, J. Hughes, and S. Lyon. 2013. Spatio-temporal indexing in non-relational distributed databases. In IEEE International Conference on Big Data. 291–299.
[95] Ellen Friedman and Kostas Tzoumas. 2016. Introduction to Apache Flink: Stream Processing for Real Time and Beyond (1st ed.). O’Reilly Media, Inc.
[96] Alessandro Furieri. 2020. SpatialLite: An Open-source Spatial Extension of SQLite. https://www.gaia-gis.it/fossil/libspatialite/index.
[97] Francisco García-García, Antonio Corral, Luis Iribarne, George Mavrommatis, and Michael Vassilakopoulos. 2017. A comparison of distributed spatial data management systems for processing distance join queries. In Advances in Databases and Information Systems. Springer, 214–228.
[98] Nandani Garg and Sayan Ranu. 2018. Route recommendations for idle taxi drivers: Find me the shortest route to a customer! In SICKDD (KDD’18). ACM, New York, NY, USA, 1425–1434.
[99] GDAL/OGR contributors. 2020. GDAL/OGR Geospatial Data Abstraction Software Library. Open Source Geospatial Foundation. https://gdal.org.
[100] Team GeoPandas. 2013–2019. GeoPandas: Python Tools for Geographic Data. https://geopandas.org/.
[101] Felix Gessert, Wolfram Wingerath, Steffen Friedlich, and Norbert Ritter. 2017. NoSQL database systems: A survey and decision guidance. Comput. Sci. 32, 3–4 (July 2017), 353–365.
[102] Sean Gillies. 2011. Fiona: Reads and Writes Geographic Data Files. The Toblerity Project. https://fiona.readthedocs.io/.
[103] Google. 2019. S2 Geometry Library. http://s2geometry.io/. Accessed on December, 2020.
[104] Noel Gorelick, Matt Hancher, Mike Dixon, Simonlyushchenko, David Thau, and Rebecca Moore. 2017. Google earth engine: Planetary-scale geospatial analysis for everyone. Remote Sensing of Environment (7 2017).
[105] Anita Graser. 2019. MovingPandas: Efficient structures for movement data in Python. Journal of Geographic Information Science 1 (6 2019), 54–68.
[106] GRASS Development Team. 2017. Geographic Resources Analysis Support System (GRASS GIS) Software, Version 7.2. Open Source Geospatial Foundation. http://grass.osgeo.org.
[107] Matthew Green. 2020. Australia’s massive fires could become routine, climate scientists warn. Reuters (2020). https://www.reuters.com/article/us-climate-change-australia-report/australias-massive-fires-could-becomeroutine-climate-scientists-warn-idUSKBN1ZD06W. January 13, 2020.
[108] Robert Grisso, Mark M. Alley, Phil McClellan, Dan Brann, and Steve Donohue. 2005. Precision farming. A comprehensive approach.
[109] Manuel Grizonnet, Julien Michel, Victor Poughon, Jordi Inglada, Mickael Savinaud, and Rémi Cresson. 2017. Orfeo ToolBox: Open source processing of remote sensing images. Open Geospatial Data, Software and Standards 2, 1 (2017), 15.
[110] Benedikt Gräler, Edzer Pebesma, and Gerard Heuvelink. 2016. Spatio-temporal interpolation using gstat. The R Journal 8 (2016), 204–218. Issue 1. https://journal.r-project.org/archive/2016/RJ-2016-014/index.html.
[111] Dongming Guo and Erling Onstein. 2020. State-of-the-art geospatial information processing in NoSQL databases. *ISPRS Int. J. of Geo-Information* 9, 5 (2020).

[112] Ralf H. Güting, Thomas Behr, and Christian Düntgen. 2010. SECONDO: A platform for moving objects database research and for publishing and integrating research implementations. *IEEE Data Eng. Bull.* 33, 2 (2010), 56–63.

[113] Ralf Hartmut Güting and Jianmin Lu. 2015. Parallel SECONDO: Scalable query processing in the cloud for non-standard applications. *SIGSPATIAL Special* 6, 2 (March 2015), 3–10.

[114] Stefan Hagedorn, Philipp Götze, and Kai-Uwe Sattler. 2017. The STARK framework for spatio-temporal data analytics on Spark. In *BTW (LNI)*, Vol. P–265. GI, 123–142.

[115] Stefan Hagedorn, Philipp Götze, and Kai-Uwe Sattler. 2017. Big spatial data processing frameworks: Feature and performance evaluation. In *EDBT*.

[116] Max Halford, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, and Adil Zoutine. 2019. Creme, a Python library for online machine learning. [https://github.com/MaxHalford/creme](https://github.com/MaxHalford/creme).

[117] David Haynes. 2019. Array databases. *GIS&T Body of Knowledge* (2019).

[118] David Haynes, Philip Mitchell, and Eric Shook. 2020. Developing the raster big data benchmark: A comparison of raster analysis on big data platforms. *ISPRS Int. J. of Geo-Information* 9, 11 (2020).

[119] David Haynes, Suprio Ray, Steven M. Manson, and Ankit Soni. 2015. High performance analysis of big spatial data. In *IEEE Big Data*. IEEE Computer Society, USA, 1953–1957.

[120] Robert J. Hjamins. 2020. terra: Spatial Data Analysis. [https://CRAN.R-project.org/package=terra](https://CRAN.R-project.org/package=terra).

[121] Robert J. Hjamins and Jacob van Etten. 2012. raster: Geographic Analysis and Modeling with Raster Data. [http://CRAN.R-project.org/package=raster](http://CRAN.R-project.org/package=raster).

[122] S. Hoyer and J. Hamman. 2017. XArray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software* 5, 1 (2017).

[123] Xuegang Huang. 2009. *Supporting Location-Based Services in Spatial Network Databases*. IGI Global, 316–324.

[124] James N. Hughes, Andrew Annex, Christopher N. Eichelberger, Anthony Fox, Andrew Hubert, and Michael Ronquest. 2015. GeoMesa: A distributed architecture for spatio-temporal fusion. In *Geospatial Informatics, Fusion, and Motion Video Analytics V*, Vol. 9473. International Society for Optics and Photonics, SPIE, 128–140.

[125] Andrey Hubert, Thomas Kunicki, James N. Hughes, Anthony D. Fox, and Christopher N. Eichelberger. 2016. An experimental study of big spatial data systems. In *2016 IEEE Big Data*. 2664–2671.

[126] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95.

[127] MongoDB Inc. 2020. MongoDB Geospatial Queries. [https://docs.mongodb.com/manual/geospatial-queries/](https://docs.mongodb.com/manual/geospatial-queries/).

[128] Neo4j Inc. 2020. Neo4j: Graphs for Everyone. [https://neo4j.com/](https://neo4j.com/).

[129] Stratio Big Data Inc. 2018. Lucene based secondary indexes for Cassandra. [https://github.com/Stratio/cassandra-lucene-index](https://github.com/Stratio/cassandra-lucene-index).

[130] Steven Johnson. 2007. A guided tour of the Ghost Map. [https://youtu.be/KvHLdHlj3RM/](https://youtu.be/KvHLdHlj3RM/).

[131] Andre Joseffson. 2018. *Comparing the Performance of Relational and Document Databases for Hierarchical Geospatial Data*. Master’s thesis. KTH Royal Institute of Technology, Stockholm, Sweden.

[132] David Kahle and H. Wickham. 2013. ggmap: Spatial visualization with ggplot2. *The R Journal* 5, 1 (2013), 144–161.

[133] Yuvaraj Sing Kanwar. 2019. *Benchmarking Scalability of NoSQL Databases for Geospatial Queries*. Master’s thesis. San Jose State University, San Jose, USA.

[134] Saadia Karim, Tariq Soomro, and S. M. Aqil Burney. 2018. Spatiotemporal aspects of big data. *Applied Computer Systems* 23 (12 2018), 90–100.

[135] S. Kim and Y. S. Kanwar. 2019. GeoYCSB: A benchmark framework for the performance and scalability evaluation of NoSQL databases for geospatial workloads. In *IEEE International Conference on Big Data*. 3666–3675.

[136] Slava Kisilevich, Florian Mansmann, Mirco Nanni, and Salvatore Rinzivillo. 2010. *Spatio-temporal Clustering*. Springer, Boston, MA, 855–874.

[137] M. Kornacker, A. Behm, V. Bittorf, T. Bobrovitsky, C. Ching, A. Choi, J. Erickson, M. Grund, D. Hecht, M. Jacobs, I. Joshi, L. Kuff, D. Kumer, A. Leblang, N. Li, I. Pandis, H. Robinson, D. Rorke, S. Rus, J. Russell, D. Tsirogiannis, and S. Wanderman-Milne, and M. Yoder. 2015. Impala: A modern, open-source SQL engine for Hadoop. In *EDBT*.

[138] Alexandre Korotkov. 2019. Viva, the NoSQL Postgres, PGCon-2019. [https://youtu.be/70dBszaO67Af](https://youtu.be/70dBszaO67Af). October, 2020.

[139] Keith Kraus. 2019. High-Performance Data Science at Scale with RAPIDS, Dask, and GPUs, PyData NYC 2019. [https://youtu.be/jjtUgsOwkl0s](https://youtu.be/jjtUgsOwkl0s). Accessed on December, 2020.

[140] Max Kuhn. 2008. Building predictive models in R using the caret package. *JSS, Articles* 28, 5 (2008), 1–26.

[141] Dymaxion Labs. 2018. Read and write rasters using Rasterio and Dask. [https://github.com/dymaxionlabs/dask-rasterio](https://github.com/dymaxionlabs/dask-rasterio).

[142] Eduardo Lacerda. 2020. Long list of Geospatial Tools and Resources. [https://github.com/sacridini/Awesome-Geospatial](https://github.com/sacridini/Awesome-Geospatial).

ACM Computing Surveys, Vol. 54, No. 10s, Article 219. Publication date: November 2022.
[174] Panagiotis Nikitopoulos, Akrivi Vlachou, Christos Doulkeridis, and George A. Vouros. 2018. DiStRDF: Distributed spatio-temporal RDF queries on Spark. In *EDBT/ICDT Workshops*.

[175] Shoji Nishimura, Sudipto Das, Divyakant Agrawal, and Amr E. Abbadi. 2011. MD-HBase: A scalable multi-dimensional data infrastructure for location aware services. In *IEEE MDM’11*, Vol. 1. 7–16.

[176] Shadi A. Noghabi, Kartik Paramasivam, Yi Pan, Navina Ramesh, Jon Bringhurst, Indranil Gupt, and Roy H. Campbell. 2017. Samza: Stateful scalable stream processing at LinkedIn. *Proc. VLDB Endow.* 10, 12 (Aug. 2017), 1634–1645.

[177] OGC. 2010. OGC Simple Feature Access - Part 2: SQL. https://www.ogc.org/standards/sfs.

[178] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. 2008. Pig Latin: A not-so-foreign language for data processing. In *SIGMOD*. ACM, New York, NY, USA, 1099–1110.

[179] Patrick O’Neil, Edward Cheng, Dieter Gawlick, and Elizabeth O’Neil. 1996. The log-structured merge-tree (LSM-tree). *Acta Inf.* 33, 4 (June 1996), 351–385.

[180] Oracle. 2019. *SQL for Oracle NoSQL Database* (v19.10).

[181] Oracle. 2020. Oracle R Connector for Hadoop (ORCH). https://docs.oracle.com/cd/E36174_01/doc.11.0.0/orch.htm.

[182] Oracle. 2020. Oracle Spatial and Graph Feature. https://www.oracle.com/database/technologies/spatialandgraph.

[183] OSGeo. 2020. GEOS - Geometry Engine, Open Source. https://trac.osgeo.org/geos.

[184] OSGeo. 2020. Open Source Geospatial Libraries. https://www.osgeo.org/choose-a-project/development/libraries/.

[185] OSGeo. 2020. Open Source Geospatial Libraries. https://www.osgeo.org/choose-a-project/development/libraries/.

[186] Nilkamal P. More, V. B. Nikam, and Sumit S. Sen. 2018. Experimental survey of geospatial big data platforms. In *IEEE HiPCW*. 137–143.

[187] Varun Pandey, Andreas Kipf, Thomas Neumann, and Alfons Kemper. 2018. How good are modern spatial analytics systems? *Proc. VLDB Endow.* 11, 11 (July 2018), 1661–1673.

[188] Varun Pandey, Alexander van Renen, Andreas Kipf, and Alfons Kemper. 2020. How good are modern spatial libraries? *DSE* (2020).

[189] Varun Pandey, Alexander van Renen, Andreas Kipf, Ibrahim Sabek, Jialin Ding, and Alfons Kemper. 2020. The case for learned spatial indexes. In *AIDB ’20*.

[190] Maria Patrou, Md Mahbub Alam, Puya Memarzadeh, Suprio Ray, Virendra C. Bhavsar, Kenneth B. Kent, and Gerhard W. Dueck. 2018. DISTIL: A distributed in-memory data processing system for location-based services. In *SIGSPATIAL*. ACM, New York, NY, USA, 496–499.

[191] Edzer Pebesma. 2012. spacetime: Spatio-temporal data in R. *JSS* 51, 7 (2012), 1–30. http://www.jstatsoft.org/v51/i07/.

[192] Edzer Pebesma. 2018. Simple features for R: Standardized support for spatial vector data. *The R Journal* 10, 1 (2018), 439–446.

[193] Edzer Pebesma. 2020. *stars: Spatiotemporal Arrays, Raster and Vector Data Cubes*. https://CRAN.R-project.org/package=stars.

[194] Edzer Pebesma, Benedikt Klus, and Mehdi Moradi. 2020. *trajectories: Classes and Methods for Trajectory Data*. https://CRAN.R-project.org/package=trajectories.

[195] Edzer J. Pebesma and Roger S. Bivand. 2005. Classes and methods for spatial data in R. *Journal of Statistical Software* 13, 9 (2005), 1–30. http://www.jstatsoft.org/v13/i09/.

[196] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[197] Nikos Pelekis, Iabas Theodoulidis, Ioannis Kopanakis, and Yannis Theodoridis. 2004. Literature review of spatio-temporal database models. *Knowl. Eng. Rev.* 19, 3 (Sep. 2004), 235–274.

[198] Oscar Perpiñan and Robert Hijmans. 2020. A Survey on Spatio-temporal Data Analytics Systems. ACM, New York, NY, USA, 45–54.

[199] Oscar Perpiñan and Robert Hijmans. 2020. A Survey on Spatio-temporal Data Analytics Systems. *Proc. VLDB Endow.* 13, 12 (Jul. 2020), 2341–2354.

[200] Meng Qu, Hengshu Zhu, Guannan Liu, and Hui Xiong. 2014. A cost-effective recommender system for taxi drivers. In *SIGKDD (KDD’14)*. ACM, New York, NY, USA, 45–54.

[201] QGIS Development Team. 2020. *QGIS: A Free and Open Source Geographic Information System*. QGIS Association. https://www.qgis.org.

[202] Jianzhong Qi, Guanli Liu, Christian S. Jensen, and Lars Kulik. 2020. Effectively learning spatial indices. *Proc. VLDB Endow.* 13, 12 (Jul. 2020), 2341–2354.

[203] Jiwei Qin, Liangli Ma, and Jinghua Niu. 2019. THBase: A coprocessor-based scheme for big trajectory data management. *Future Internet* 11, 1 (2019).

[204] R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.
[205] Paul Ramsey. 2019. Parallel PostGIS and PgSQL 12. http://blog.cleverelephant.ca/2019/05/parallel-postgis-4.html.

[206] Suprio Ray, Bogdan Simion, Angela Denke Brown, and Ryan Johnson. 2013. A parallel spatial data analysis infrastructure for the cloud. In SIGSPATIAL. ACM, New York, NY, USA, 284–293.

[207] RedisLabs 2020. Redis: An in-memory database that persists on disk. https://redis.io/. Accessed on June, 2020.

[208] RedisLabs. 2020. Redis Geospatial Support. https://redislabs.com/redis-best-practices/indexing-patterns/geospatial/.

[209] Data Reply. 2017. Benchmarking of big data technologies for ingesting and querying geospatial datasets. (2017).

[210] Sergio Rey, Luc Anselin, Xun Li, Robert Pahle, Jason Laura, Wenwen Li, and Julia Koschinsky. 2015. Open geospatial analytics with PySAL. ISPRS International Journal of Geo-Information 4, 2 (May 2015), 815–836.

[211] Sergio J. Rey and Luc Anselin. 2007. PySAL: A Python library of spatial analytical methods. The Review of Regional Studies 37, 1 (2007), 5–27.

[212] RHadoop 2011. RHadoop. https://github.com/RevolutionAnalytics/RHadoop. Last Accessed June, 2020.

[213] Ramon Antonio Zalipynis. 2019. ChronosDB in action: Manage, process, and visualize big geospatial arrays in the cloud. In Proceedings of the 2019 International Conference on Management of Data (SIGMOD’19). ACM, New York, NY, USA, 1985–1988.

[214] David S. Rosenberg. 2012. HadoopStreaming: Utilities for using R scripts in Hadoop streaming. https://CRAN.R-project.org/package=HadoopStreaming.

[215] S. Ruan, R. Li, J. Bao, T. He, and Y. Zheng. 2018. CloudTP: A cloud-based flexible trajectory preprocessing framework. In ICDE. 1601–1604.

[216] Jeffrey A. Ryan and Joshua M. Ulrich. 2020. xts: eXtensible Time Series. https://CRAN.R-project.org/package=xts.

[217] Henk Scholten, Steven Frujtier, Arta Dilo, and Erik Borkulo. 2008. Spatial Data Infrastructure for Emergency Response in Netherlands. 179–197.

[218] Skipper Seabold and Josef Perktold. 2010. statsmodels: Econometric and statistical modeling with Python. In 9th Python in Science Conference.

[219] Team Seaborn. 2017. mwaskom/seaborn: v0.8.1 (September 2017).

[220] Salman Ahmed Shaikh, Komal Mariam, Hiroyuki Kitagawa, and Kyoung-Sook Kim. 2020. GeoFlink: A distributed and scalable framework for the real-time processing of spatial streams (CIKM’20). ACM, New York, NY, USA, 8.

[221] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: A distributed in-memory trajectory analytics system. In SIGMOD. ACM, New York, NY, USA, 1681–1684.

[222] Team Shapely. 2013. Shapely: Manipulation and Analysis of Geometric Objects in the Cartesian Plane. The Toblerity Project. https://shapely.readthedocs.io/.

[223] Shashi Shekhar, Zhe Jiang, Reem Y. Ali, Emre Efteiloglu, Xun Tang, Venkata M. V. Gunturi, and Xun Zhou. 2015. Spatiotemporal data mining: A computational perspective. ISPRS Int. J. of Geo-Information 4, 4 (2015), 2306–2338.

[224] Justin Sheng and Rüdiger Busche. 2019. justinshenk/traja: v0.1.1.

[225] Carson Sievert. 2020.

[226] Julia Signell. 2020. ParallelGeoPandaswithDASK.

[227] John Snow Society. 2020. John Snow: The Pioneer of Epidemiological Method and Celebrated Anaesthetist. http://www.johnsnowsociety.org/.

[228] Chronicle Software. 2021. Chronicle Map: A High Performance, Off-heap, Key-value, In-memory, Persisted Data Store. Retrieved January 2021 from https://github.com/OpenHFT/Chronicle-Map.

[229] Ram Sriharsha. 2017. Magellan: Geospatial Data Analytics on Spark. https://github.com/harsha2010/magellan.

[230] Knut Stolze. 2003. SQL/MM spatial: The standard to manage spatial data in relational database systems. In BTW. Gesellschaft für Informatik e.V., Bonn, 247–264.

[231] Michael Stonebraker, Paul Brown, Alex Poliakov, and Suchi Raman. 2011. The architecture of SciDB. In SSDBM. Springer-Verlag, Berlin, 1–16.

[232] Rob Story. 2013. Folium: Python data, leaflet.js maps. https://python-visualization.github.io/folium/.

[233] Dain Sundstrom. 2020. LevelDB Java Version. https://github.com/dain/leveldb. Last Accessed June, 2020.

[234] Mingjie Tang, Yongyang Yu, Walid G. Aref, Ahmed R. Mahmood, Qutaibah M. Malluhi, and Mourad Ouzzani. 2019. LocationSpark: In-memory distributed spatial query processing and optimization. arXiv e-prints (Jul. 2019).

[235] Mingjie Tang, Yongyang Yu, Qutaibah M. Malluhi, Mourad Ouzzani, and Walid G. Aref. 2016. LocationSpark: A distributed in-memory data management system for big spatial data. VLDB Endow. 9, 13 (Sept. 2016), 1565–1568.

[236] Couchbase Team. 2011. Couchbase: A Distributed NoSQL Document Database. https://www.couchbase.com/.

[237] Numpy Team. 2020. Array programming with NumPy. Nature 585 (2020), 357–362.

[238] PL/Proxy Team. 2020. PL/Proxy: Function-based Sharding for PostgreSQL. https://plproxy.github.io/.

[239] PostgreSQL Team. 2018. PostgreSQL: A Scalable PostgreSQL-based Database Cluster. https://www.postgres-xl.org/.

[240] Rasterio Team. 2018. Rasterio: Access to Geospatial Raster Data. https://github.com/mapbox/rasterio/.

[241] RAPIDS Development Team. 2018. RAPIDS: Collection of Libraries for End to End GPU Data Science. https://rapids.ai.

[242] SAGA Team. 2001. SAGA: System for Automated Geoscientific Analyses. http://www.saga-gis.org/. June, 2020.
A Survey on Spatio-temporal Data Analytics Systems

[243] SciPy Team. 2020. SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods (2020).

[244] Martijn Tennekes. 2018. tmap: Thematic maps in R. JSS 84, 6 (2018), 1–39.

[245] Aman Thakral. 2016. pyspatial: Data Structures on Top of GIS/OGC. https://pyspatial.readthedocs.io/.

[246] The Luigi Authors. 2020. Luigi: A Python Module with Hadoop Supports. https://luigi.readthedocs.io/.

[247] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. 2009. Hive: A warehousing solution over a map-reduce framework. Proc. VLDB Endow. 2, 2 (Aug. 2009), 1626–1629.

[248] Finbar Tierman and Eamonn O’Mallon. 2020. Australia’s 2019–20 bushfire season. The Canberra Times (2020). https://www.canberratimes.com.au/story/6574563/australias-2019-20-bushfire-season/, January 13, 2020.

[249] Luis Torgo. 2014. An infra-structure for performance estimation and experimental comparison of predictive models in R. CoRR abs/1412.0436 (2014).

[250] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and contributors. 2014. scikit-image: Image processing in Python. PeerJ 2 (6 2014).

[251] Guido Van Rossum and Fred L. Drake Jr. 1995. Python Tutorial. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

[252] Shivaram Venkataraman, Zongheng Yang, Davies Liu, Eric Liang, Hossein Falaki, Xiangrui Meng, Reynold Xin, Ali Ghodsi, Michael Franklin, Ion Stoica, and Matei Zaharia. 2016. SparkR: Scaling R programs with spark. In SIGMOD’16. ACM, New York, NY, USA, 1094–1109.

[253] Hoang Vo, Ablimit Aji, and Fusheng Wang. 2014. SATO: A spatial data partitioning framework for scalable query processing. In SIGSPATIAL’14. ACM, New York, NY, USA, 545–548.

[254] Tin Vu and Ahmed Eldawy. 2020. R*-grove: Balanced spatial partitioning for large-scale datasets. Frontiers in Big Data 3 (2020), 28.

[255] Leye Wang and O. Wolfson. 2020. Grand challenges for the spatial information community. JOSIS 20 (2020), 79–85.

[256] Alex Watson, Suvam Kumar Das, and Suprio Ray. 2021. DaskDB: Scalable data science with unified data analytics and in situ query processing. In IEEE 8th DSA’2021. 1–10.

[257] Wes McKinney. 2010. Data structures for statistical computing in Python. In 9th Python Science Conference. 56–61.

[258] Jeffrey Whitaker. 2018. pyproj: Python Interface to PROJ. https://pyproj4.github.io/pyproj/latest/.

[259] Randall T. Whitman, Michael B. Park, Sarah M. Ambrose, and Erik G. Hoel. 2014. Spatial indexing and analytics on Hadoop. In SIGSPATIAL. ACM, New York, NY, USA, 73–82.

[260] Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. 2020. dplyr: A Grammar of Data Manipulation. https://CRAN.R-project.org/package=dplyr.

[261] Hadley Wickham and Edgar Ruiz. 2020. dbplyr: A ‘dplyr’ Back End for Databases. https://CRAN.R-project.org/package=dbplyr.

[262] Dawn J. Wright. 2013. The Ocean GIS Initiative. ESRI Publications.

[263] Dong Xie, Feifei Li, Bin Yao, Gefei Li, Zhongpui Chen, Liang Zhou, and Minyi Guo. 2016. Simba: Spatial in-memory big data analysis. In 4th ACM SIGSPATIAL. ACM, New York, NY, USA, Article 86, 4 pages.

[264] Dong Xie, Feifei Li, Bin Yao, Gefei Li, Liang Zhou, and Minyi Guo. 2016. Simba: Efficient in-memory spatial analytics. In SIGMOD’16. ACM, New York, NY, USA, 1071–1085.

[265] Zhong Yang, Bolong Zheng, Chengdong Tong, Lianggui Weng, Chenliang Li, and Guohui Li. 2021. HASTE: A Distributed System for Hybrid and Adaptive Processing on Streaming Spatial-Textual Data. ACM, 2363–2372.

[266] Xiaochuang Yao and G. Li. 2018. Big spatial vector data management: A review. Big Earth Data 2, 1 (2018), 108–129.

[267] Yel and Contributors. 2018. mrjob: Run MapReduce Jobs on Hadoop. https://mrjob.readthedocs.io/.

[268] S. You, J. Zhang, and L. Gruenwald. 2015. Large-scale spatial join query processing in cloud. In IEEE International Conference on Data Engineering Workshops. 34–41.

[269] Jia Yu, Jinxuan Wu, and Mohamed Sarwat. 2015. GeoSpark: A cluster computing framework for processing large-scale spatial data. In SIGSPATIAL’15. ACM, New York, NY, USA, Article 70, 4 pages.

[270] Jia Yu, Zongzi Zhang, and Mohamed Sarwat. 2019. Spatial data management in Apache Spark: The GeoSpark perspective and beyond. Geoinformatica 23, 1 (Jan. 2019), 37–78.

[271] Jing Yuan, Yu Zheng, and Xing Xie. 2012. Discovering regions of different functions in a city using human mobility and POIs. In SIGKDD (KDD’12). ACM, New York, NY, USA, 186–194.

[272] Jing Yuan, Yu Zheng, Lithuan Zhang, Xing Xie, and Guangzhong Sun. 2011. Where to find my next passenger. In UbiComp. ACM, New York, NY, USA, 109–118.

[273] Nicholas J. Yuan, Yu Zheng, Liuhang Zhang, and Xing Xie. 2015. Discovering urban functional zones using latent activity trajectories. TKDE 27, 3 (2015), 712–725.

[274] Nicholas J. Yuan, Yu Zheng, Liuhang Zhang, and Xing Xie. 2012. T-finder: A recommender system for finding passengers and vacant taxis. TKDE (Sep. 2012).

ACM Computing Surveys, Vol. 54, No. 10s, Article 219. Publication date: November 2022.
Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *USENIX (NSDI’12)*. USENIX Association, San Jose, CA, 15–28.

Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster computing with working sets. In *USENIX (HotCloud’10)*. USENIX Association, USA, 10.

Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shvaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache spark: A unified engine for big data processing. *Commun. ACM* 59, 11 (Oct. 2016), 56–65.

Ramon Antonio Rodriges Zalipynis. 2018. ChronosDB: Distributed, file based, geospatial array DBMS. *Proc. VLDB Endow.* 11, 10 (Jun. 2018), 1247–1261.

Ramon Antonio Rodriges Zalipynis. 2021. Array DBMS: Past, present, and (near) future. *Proc. VLDB Endow.* 14, 12 (Jul. 2021), 3186–3189.

Achim Zeileis and Gabor Grothendieck. 2005. zoo: S3 infrastructure for regular and irregular time series. *JSS* 14, 6 (2005), 1–27.

Yaming Zhang and Ahmed Eldawy. 2020. Evaluating computational geometry libraries for big spatial data exploration (*GeoRich’20*). ACM, New York, NY, USA, Article 3, 6 pages.

Zhigang Zhang, Cheqing Jin, Jiali Mao, Xiaolin Yang, and Aoying Zhou. 2017. TrajSpark: A scalable and efficient in-memory management system for big trajectory data. In *Web and Big Data*. Springer, 11–26.

Yu Zheng. 2015. Trajectory data mining: An overview. *ACM TIST* 6, 3, Article 29 (May 2015), 41 pages.

Esteban Zimányi, Mahmoud Sakr, Arthur Lesuisse, and Mohamed Bakli. 2019. MobilityDB: A mainstream moving object database system. In *SSTD*. ACM, 206–209.

Received 9 March 2021; accepted 20 December 2021