Comparative Analysis of Spark and Ignite for Big Spatial Data Processing

Samah Abuayeid, Louai Alarabi
Department of Computer Science
Umm Al-Qura University
Mecca, Saudi Arabia

***Abstract*—Recently, spatial data became one of the most interesting fields related to big data studies, in which the spatial data have been generated and consumed from different resources. However, the increasing numbers of location-based services and applications such as Google Maps, vehicle navigation, recommendation systems are the main foundation of the idea of spatial data. On the other hand, several researchers started to discover and compared spatial frameworks to understand the requirements for spatial database processing, manipulating, and analysis systems. Apache Spark, Apache Ignite, and Hadoop are the most widely known frameworks for large data processing. However, Apache Spark, Apache Ignite have integrated different spatial data operations and analysis queries, but each system has its advantages and disadvantages when dealing with spatial data. Dealing with a new framework or system that needs to integrate new functionality sometimes becomes a risky decision if we did not examine it well The main aim of this research is to conduct a comprehensive evaluation of big spatial data computing on two well-known data management systems Apache Ignite and Apache Spark. The comparative has been done on four different domains, experimental environment setup, supported features, supported functions and queries, and performance and execution time. The results show that GeoSpark has recorded more flexibility to use than SpatialIgnite. We thoroughly investigated and discovered that multiple factors affect the performance of both frameworks, such as CPU, Main memory, data set size the complexity of data type, and programming environment. spark is more advanced and equipped with several functionalities that made it well suitable with spatial data queries and indexing. such as kNN queries; in which these functionalities are not supported in SpatialIgnite.**

***Keywords*—Big spatial data; GeoSpark; SpatialIgnite; Apache Ignite; Apache Spark**

I. Introduction

Big data processing has always been a critical research area in both academia and industry. Several big tech organizations invested billions of dollars to build Big data Eco-system, For example, Facebook [1], LinkedIn [2], Microsoft [3], ESRI [4] to name a few. Meanwhile, several non-tech companies have integrated one or more available platforms to scale out and perform their big data analytic tasks. One important domain of this market is building Eco-systems for spatial data due to the plethora of applications and services that create them. For instance, earth observation has continuously provided a significant volume of geospatial data over the last few years, resource tracking [5], environmental protection, and disaster predictions [6]. Thus, big data spatial computing has become extremely valuable with the widespread use of these services and applications.

The growth market and data size of location-based services contributed to the advancement and complexity of computing spatial data [7]. Several efforts from both industry [8] and academia [9], [10], [11], [12] introduced a specialized spatial data processing engine to process this complex data. The developments of specialized geospatial processing engines are driven by adopting new technologies for processing big data. An essential aspect of any spatial system is how the system deals with the Big V's such as Volume, Variety, veracity, and Velocity. For instance, several research studies extended these frameworks to specific domain applications, such as Transportation [13], [14], [15], [16]

Although many systems have been established to leverage the processing of spatial data. Yet, there is no single source of a comprehensive benchmark that distinguishes between these systems. The lack of possessing this kind of benchmark is due to the complexity and efforts associated with building them up. On the other hand, the variety of spatial data types in different domains (e.g., location, routing, navigation) makes it even harder to benchmark. Thus, we extensively spend a decent effort designing and assisting the performance of well-known big in-memory spatial platforms built on Apache Spark and Apache Ignite.

Apache has founded and managed several big data processing projects [17], such as Hadoop [18], Spark [19] and Ignite [20] among others. This paper, investigated Spark and Ignite which are commonly known as distributed in-memory big data processing platforms. Researchers investigated these two platforms in processing big spatial data, by introducing and building spatial properties, operations, and queries in these two platforms. For instance, Apache Spark [21] was introduced as the GeoSpark system, in which users can interacting with the system by either: Spatial SQL API or a Scala/Java RDD API. spark provides for the users an operational programming language for writing a custom spatial analytic application. On the other hand, Apache Ignite [22] is another open-source distributed database system that includes an in-memory data grid (IMDG) that was established to store and compute big data across a cluster of nodes. However, to integrate the spatial data processing and quires in Apache Ignite, users must add the dependency of an ignite-geospatial library that is included in the JTS Topology Suite.

Dealing with a new framework or system that needs to integrate new functionality sometimes becomes a risky decision if we did not examine it, and In the literature up to now, there are few comparative studies between Apache Spark and Apache Ignite systems in spatial data management domain.

This paper, motivated by a prior study comprehensively investigated and evaluated a native version of Apache Hadoop and Spark, respectively [23]. In this research the study took the leverage of the advancement of the field of spatial data computing and comprehensively evaluate and compare two distributed in-memory computing frameworks and well-known data management frameworks of Apache Ignite and Apache Spark. In particular, the study evaluated the spatial extension of Apache Spark that is well-known as GeoSpark with it's competitor SpatialIgnite on Apache ignite.

The rest of this paper has organized as follows. Section 3 research background, which has divided into two subsections. Section 4 consists of materials and methods, which has divided into two subsections. Section 5 included the results and discussion. Concluding remarks are contained in the last section.

II. Related Work

New research by Cristiana-Stefania Stan [24] has compared Apache Spark and Apache Ignite from different aspects such as implementation, features, architecture, and performance metrics using word count and k-means clustering to specify the best framework for Big data processing. The experiment was conducted on two nodes to offer a small cluster, which made the testing step accurate because it can measure the communication between the nodes. On the other hand, they also test the computation duration and multiple systems resources utilization such as CPU, memory, and network, as a result the author diseased that Apache Spark has addressed a higher performance in processing Big data than Apache Ignite.

Md Mahbub Alam [12] has studied the performance of Big spatial data on Apache Spark, Spatial-Ignite benchmark developed based on Apache Ignite and HADOOP. The author has realized that not all the systems have supported all Spatial Data features. The work has inspired form [33], which proposed the d Jackpine spatial database benchmark to support as many as possible spatial databases with minimal effort. However, the experiment in [12] has been conducted using a real-world spatial dataset from TIGER on a cluster of 8 machines with the two frameworks (Apache Spark and Hadoop) and the proposed benchmark (Spatial-Ignite). The result has addressed three categories of the operations for examination: spatial join, spatial analysis queries, and range query. At the end of the experiment, Spatial-Ignite performs better than Apache Spark and Hadoop on the spatial queries.

Recently, social media platforms provided a huge amount of spatial and temporal datasets, and several spatial data processing frameworks were developed to dealing with this type of massive data. To illustrate this, the study by Zhibo Sun [25], investigated Apache Spark to perform some geospatial analyses operations, such as K-Nearest Neighbors (KNN), the distribution of the median points, and the geographic mean and median points. The study applied three different sizes of Twitter data: 180GB, 120GB, and 60GB. Additionally, they compared the execution results and performance of Apache Spark with Hadoop, in which Spark addressed a high performance than Hadoop when applied to the Twitter dataset. Furthermore, the experiment was conducted on the Amazon EC2 cluster, that consists off 11 m3.xlarge nodes. However, each instance in the m3.xlarge has "Intel Xeon E5-2670 v2, 4 vCPU, 15 GB memory, and 100 GB magnetic storage", where the study was dependent on Apache Hadoop 2.4, Spark 1.1.1. By comparing Hadoop and Spark in computing the distribution of the median points, and geographic mean, Spark addressed better performance than Hadoop, in which writing and reading data in and out of the disk take more time with the Hadoop solution. On the other hand, Spark also records a high performance faster than Hadoop, equals 2.3x, 1.6x, and 1.8x on the three diffident sizes of Twitter datasets. As a result Spark solution was outperformed the Hadoop solution, but the author was found some disadvantages and limitations in the Spark-based solution:

- Spark uses some coarse-grained mode, that enhance their performance, but on the other hand it wastes lots of resources, which cause jobs delay.
- Spark uses a lazy operation mechanism, that requires a run of some actions before debugging.
- Spark can cost on physical nodes, in which it needs large memory in compared to Hadoop.

Moreover, Randall T. Whitman [26] was studied the spatial join operations on Apache Spark, and proposed a framework that uses the spatiotemporal join algorithms using two datasets. The framework is expected to runs in commercial off-the-shelf (COTS) applications. The study has employed two approaches to perform the spatial join operation. The first approach was the broadcast spatial join, that built to join a big spatial dataset with another small dataset, where, the second approach was the Bin spatial join, which is a technique for joining two large datasets. However, the broadcast join is similar to the map-side join in MapReduce programming, and it is suitable when one of two datasets can fit into memory on the Spark executors, so the study used this join operation to determine which one of the dataset is smaller and can fit be into the Spark memory. On the other hand, Bin Join is similar to the reduce-side join in MapReduce programming, and it is usually convenient when there is a need for a partitioned approach because the datasets are too large to fit into memory. The study was started by defining the size of the two datasets, and they were distributed into Spark executors, then the datasets partitions in each executor were joining using the broadcast join. The next step was using the bin join, and the same binning operation was applied on the two datasets, to ensure that the features on each side of the spatial grid are the same. In the end, the de-duplication step was performed to remove the duplicate matches when the features become in multiple bins. However, to estimate the proposed framework performance the study was conducted on New York City Taxi and Limousine Commission's taxi dataset, and they concluded, that join algorithms depend on the characteristics of the two datasets. On the other hand, they observed that the broadcast join is faster when one of the datasets is in a modest size, and the operation performance decreased when the two datasets are large. Moreover, the fastest binning technique is the regular grid mesh.

On the other hand, Jia Yu [27] was presented the GeoSpark framework, which is an in-memory cluster computing framework for manipulating large spatial data. The proposed framework consists of three main layers: Apache Spark layer,

TABLE I. LITERATURE REVIEW SUMMARY

Paper	Author	Year	Framework	Result
[24]	Cristiana-Stefania	2019	Spark and Ignite	Spark addressed higher performance
[12]	Md Mahbub	2018	Spark, Spatial-Ignite, and Hadoop	Spatial-Ignite addressed higher performance than Spark and Hadoop
[25]	Zhibo Sun	2016	Spark and Hadoop	Spark outperformed Hadoop
[26]	Randall T. Whitman	2017	Spark spacial join algorithms	broadcast join address best performance
[27]	Jia Yu	2019	GeoSpark and Hadoop	GeoSpark outperform Hadoop
[28]	Jayati Gandhi	2020	GeoSpark and Spatial Hadoop	GeoSpark more efficient than Spatial Hadoop
[29]	Mingjie Tang	2016	Location-Spark performance	addressed good performance in spatial queries processing
[30]	Francisco Garcıa	2017	spatialHadoop and LocationSpark	LocationSpark outperformed spatialHadoop
[31]	Zhou Huang	2017	GeoSpark SQL, PostGIS, and Hadoop ESRI	GeoSpark SQL was more user friendly
[32]	Panagiotis Nikitopoulos	2018	DiStRDF under SPARK	process RDF spatial temporal queries in minimum time
[10]	Louai Alarabi	2018	ST-Hadoop	ST-Hadoop outperformed Hadoop and SpaitalHadoop

spatial RDD layer, and the spatial query layer, the framework allows the user to built a spatial index. However, each layer is responsible for specific functionality in the framework: the Apache Spark layer responsible for providing the basic Spark functionality as loading and sorting data in the disk, where the second layer the spatial RDD layer can support the geometrical, and spatial objects. On the other hand, the spatial query layer's main functionality is to execute the main spatial query processing algorithms such as KNN, Join, and spatial range. Moreover, the study was compared the performance of the proposed framework with Hadoop spatial computation. The study was conducted on three different datasets from TIGER files: Zcta510 1.5 GB dataset, Areawater 6.5 GB dataset, and Edges 62 GB dataset. The experiment was started by configuration setup: one CPU for each worker, two Memory per worker each one consist of 61 GB, 50 GB registered memory in Spark and Hadoop and three storage per worker. After that they studied the time performance of some large spatial data processing systems: GeoSpark-NoIndex, Quad-Tree, RTree, GeoSpark without spatial index, and with spatial Quad-Tree or R-Tree index, then the performance was compared with spatial Hadoop. The experiment concludes with that GeoSpark record a better performance than Spatial Hadoop, and as a future work GeoSpark can be used in multiple fields and different users such as space scientists, geographers, politicians, commercial institutions to support the spacial data analyzing process.

On the other hand, a study by Jayati Gandhi [28] was discussed the importance of geospatial data processing and analysis, which lead to the discovery of various spatial data frameworks. The study also presented a comparative study between GeoSpark and Spatial Hadoop, in which they are two of the most known open source geospatial big data analytic frameworks. The study was compared between the two frameworks according to multiple characteristics such as the architecture View, spatial data processing approach, and real-time performance of each framework. As a result, the author was found, that both frameworks: GeoSpark and Spatial Hadoop are suitable and flexible to dealing with geospatial data, but the experiment was recorded, that GeoSpark is more efficient and fast than Spatial Hadoop. In future work, the author was mentioned that they tend to use the study result to enhance the way of disaster management, and geospatial health infrastructure.

Another study in the research area was presented by Mingjie Tang [29] discussed the efficiency of MapReduce-based systems, in which it allows performing spatial queries using predefined spatial operations without the need to worry about fault tolerance and computation distribution problems. However, MapReduce-based systems had addressed some disadvantages such as, the lack of leverage distributed memory, and they do not allow immediate data reuse, in which data reusing is very common in spatial data preprocessing. Consequently, the study has introduced a solution that can fix these problems by built a spatial data processing system on Apache Spark and known as the Location-Spark. The system consists of 6 different layers: memory management, spatial index, query executor, query scheduler, spatial operator, and spatial analysis. Moreover, Location-Spark allows users to use a set of spatial query operators, such as range search, spatial-join, KNN, spatial-textual operation, and kNN-join. Moreover, the proposed system was offered multiple in-memory data spatial indexes, and guaranteed fixed spatial indexes decreased the overhead of fault tolerance. As a result, Location-Spark was outperformed in speed and performance of other spatial systems.

A study by Francisco Garcıa [30] was compared between spatialHadoop and LocationSpark by estimating the performance of each spatial data system in the way that they

performed two spatial join queries: K Closest Pair Query (KCPQ) and the ϵ Distance Join Query. The study was conducted on Linux operating system using three 2d point spatial datasets from Open-Street-Map data: building dataset includes 115M records, lakes dataset contains 8.4M points, and parks consist of 10M records. The experiment was implemented the KCPQ and ϵ Distance, Join Query, on the three datasets on spatialHadoop and LocationSpark. The study has adopted the idea of plane sweep techniques. On the other hand, The performance was evaluated depending on n the total execution time. As a result, the author found that LocationSpark was addressed the best performance comparing to spatialHadoop, according to the in-memory processing and the query plan scheduler offered by Spark. Additionally, the advantage of SpatialHadoop appeared in it is had more mature and robust DSDMS than Spark. However, as a future work, the paper tended to apply a new study on Spark-based DSDMS, such as Simba, and apply some spatial partitioning techniques on LocationSpark.

Moreover, Zhou Huang [31] was discussed the requirements of spatial data and spatial data query processing in the area of big data. Additionally, the study was introduced the GeoSpark SQL, which is a framework that allows the operations of spatial data query processing on Apache Spark. Furthermore, the paper was addressed the main issues related to introducing spatial big data query processing: storage management methods, spatial operations implementation approach, and spatial query optimization approaches. The author was chosen Apache spark in this study rather than Hadoop, in which that spark had the ability of efficient memory management, that enhance the computations of MapReduce. The experiment was covered multiple spatial queries such as KNN query, point query, window query, range query, directional query, topological query, and multi-table spatial join query. However, for experiment evaluation 10 different test cases were created, and the average time was taken as a performance evaluation metric, for comparing the performance of GeoSpark SQL with PostGIS, and Hadoop ESRI Spatial Framework. As a result, GeoSpark SQL was user-friendly, and it only need to add Java dependencies for spatial data query processing in Spark, and then start spark terminal. As future work, the author tended to improve the environment of GeoSpark SQL to integrate complex spatial data indexes.

Moreover, Panagiotis Nikitopoulos in his research[32] proposed the DiStRDF system using the resource description framework, which is a framework for web data modeling and interchanging. Additionally, the study has used Spark as the processing framework. Moreover, the experiment introduced a challenging solution for interlinking and interchanging the spatial-temporal data obtained from mobile devices. Moreover, the proposed system has addressed a good performance in the process of the resource description framework spatial-temporal queries with minimum execution time.

On the other hand, Alarabi in his research [10] proposed an open-source MapReduce framework, which supports the spatial-temporal data known as ST-Hadoop. However, the proposed framework contained one primary node, which helps in breaking map-reduce jobs into multiple tasks. Additionally, three different users are allowed by the system: casual users, developers, and administrators. According to the author, the framework is divided into four layers: language layer, indexing layer, MapReduce layer, and operations layer. The experiment was conducted on a dataset with over 1 Billion spatial-temporal data of size 10 TB. As a result, ST-Hadoop outperformed Hadoop and SpaitalHadoop in processing spatial-temporal data. However, Table I shows a summary for this section.

III. Materials and Method

This section introduce the research methodology in more detail with a clear explanation for the dataset, tools, and experiment environment that have been used to conduct the study. However, the experiment was conducted on one machine a HP Pavilion laptop with 8GB RAM, 1 TB HDD storage with windows 10 as an operating system. The research methodology as it shown in Fig. 1 started by defined the comparative study four main domains, which are experimental environment setup, supported features, supported functions and queries, and performance and execution time. Additionally, in the last step of comparison we have selected multiple datasets and calculated the execution time. However, this research is inspired by Md Mahbub Alam [12], in which it was depend on some assumptions and results obtained by Md Mahbub Alam.

A. Data

In this section we have explained the dataset used for study the performance and execution time for GeoSpark on Apache Spark, to understand the assumption that has been proposed by Md Mahbub Alam [12], in which that GeoSpark recorded the worst performance and execution time in comparing to SpatialIgnite and SaptialHadoop. However, we have utilizes real-world spatial datasets which were obtained from the Spatial Hadoop framework website[1]. However, the Spatial Hadoop website provided multiple TIGER 2015 spatial datasets and Open Street Map datasets. TIGER datasets were extracted from the US Census Bureau TIGER files. On the other hand, Open Street Map is a collaborative project to create a free editable map of the world and offered an API for data extraction. Two different TIGER datasets were chosen in this research: AREALM dataset 140MB, and PRIMARY ROADS dataset 52 MB. However, we have used two points CSV datasets 487 KB and 325 KB, which was been obtained from SpiderWeb[2]. The SpiderWeb is a website that generates spatial data, in which users can easily choose the number of layers, Cardinality, file extensions, and other proprieties. In the end, we have applied four different types of datasets on Apache Spark, Table II shows some information about our experiment datasets. However, all datasets have one attribute as the coordinate attribute.

B. The Proposed Study among Apache Spark and Apache Ignite

The study in this research was divided into four parts as follows:

1) A comparison among experimental environment setup: explained the main requirements and steps

[1]http://spatialhadoop.cs.umn.edu/datasets.html
[2]https://spider.cs.ucr.edu/

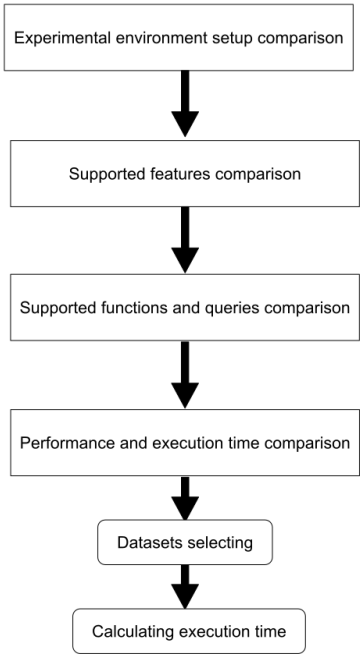

Fig. 1. The Research Methodology.

TABLE II. DATASETS DESCRIPTION

Dataset	Geometry	Cardinality	Size	Extension	Description
AREALM	Polygon	129K	140MB	CSV	Area Landmark
PRIMARYROADS	Linestrings	12,396	52 MB	CSV	Primary Roads
data1	points	12,000	487 KB	CSV	points data from SpiderWeb
data2	points	8000	325 KB	CSV	points data from SpiderWeb

need it for each framework to activate the big spatial data computing.

2) A Comparison among main supported features in each framework: explained the main supported input format, geometry type, query language, and more.
3) A comparison among the existing supported functions and queries.
4) Discussing the performance and execution time in each framework, and we have depended on the assumption proposed by Md Mahbub Alam [12], in which SpatialIgnite have the best performance, for that we have focused on calculating the performance time for Geospark using multiple datasets to understand the performance rate of GeoSpark.

1) Experimental Environment Setup: To activate the spatial computing option. The experiment was conducted using two Apache software: Apache Ignite 2.8.0[3], Apache Spark 3.0.1[4]. However, Apache NetBeans IDE 11.3 as an IDE environment was installed for working with Apache Ignite, in which the only way to support the spatial data operations and quires is by adding the Spatial Ignite dependencies on Maven in any Java IDE. Additionally, Spatial Ignite is an API of spatial predicates in the JTS Topology Suite[5]. On the other hand, Apache Spark was installed to add the GeoSpark dependency for working with GeoSpark on Apache Spark. Additionally, the study was conducted on Geospark rather than Spacial Spark because GeoSpark offered more file extensions and allow spatial indexing. However, the main programming environment that was used was Scala on the Spark shell, and the main experimental environment setup for the framework shows in Fig. 2.

2) Supported Features: This part presented a comparison between GeoSpark and SpatialIgnite among their main features. According to Table III, GeoSpark and SpatialIgnite supported the same geometry type: point, line, and polygon. Additionally, spatial analysis is supported only by SpatialIgnite. However, GeoSpark has the advantage of supporting multiple file formats: CSV, TSV, WKT, and GeoJSON, where SpatialIgnite only supports WKT file format, which decreases the flexibility of SpatialIgnite. On the other hand, Table III, shows that SpatialIgnite supports the distributed SQL query language, where GeoSpark supports SQL 2017, which may make GeoSpark easier than SpatialIgnite in the way of handling SQL queries. Another advantage for GeoSpark is that it supports two types of indexing, which are the R-tree and Quadtree, where SpatialIgnite only supports the R-tree

[3]https://ignite.apache.org/
[4]https://spark.apache.org/
[5]http://www.tsusiatsoftware.net/jts/main.html

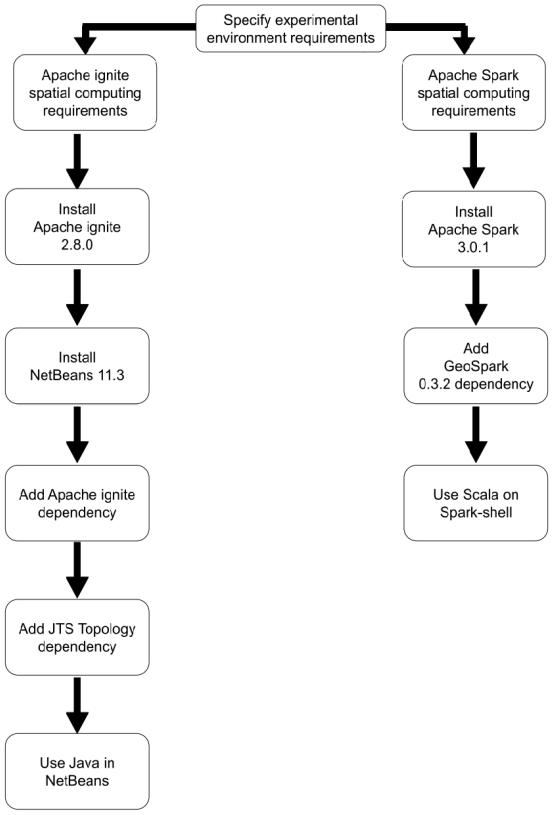

Fig. 2. The Experimental Environment Setup.

TABLE III. SUMMARY FOR THE SUPPORTED FEATURES IN GEOSPARK AND SPATIAL IGNITE

Features	GeoSpark	Spatial Ignite
geometry type	point, line, polygon	point, line, polygon
input format	CSV,TSV,WKT,GeoJSON	WKT
query language	SQL(2017)	Distributed SQL
indexing	R-tree, Quad-tree	R-tree
spatial analytic	No	Yes

indexing.

TABLE IV. SUMMARY OF SOME SUPPORTED SPATIAL FUNCTIONS AND QUERIES IN GEOSPARK AND SPATIAL IGNITE

Predicates	GeoSpark	Spatial Ignite
Equals	Yes	Yes
Intersects	Yes	Yes
Crosses	No	Yes
Envelope	Yes	Yes
ConvexHull	No	Yes
Within	No	Yes
Touches	Yes	Yes
KNN-query	Yes	No

3) Supported Spatial Functions and Queries: Table IV, shows a summary for some spatial functions and queries supported by GeoSpark and SpatialIgnite. Table IV confirms the existence of some similarities between GeoSpark and SpatialIgnite in the supported spatial predicates, in which that GeoSpark and SpatialIgnite are supported for the same predicates, such as Equals, Intersects, Envelope, Touches. On the other hand, KNN-query is only supported by GeoSpark, where Range-query and Join-query are supported by the two frameworks (GeoSpark and SpatialIgnite). However, from Table IV we have observed that SpatialIgnite supports more spatial functionality than GeoSpark, but GeoSpark in Apache Spark is more suitable with spatial data queries and indexing because it support complex spatial queries, such as kNN queries, which is not supported in SpatialIgnite on Apache ignite. Moreover, GeoSpark is an agile spatial data processing framework that can meet the changes in the requirements, and fault tolerance.

4) Performance and Execution Time: According to the results and assumptions produced by Md Mahbub Alam [12], which proposed that SpatialIgnite has addressed the best performance among SaptialHadoop and GeoSpark, and from this research observation GeoSpark has been widely used for big spatial data processing, and many spatial frameworks was been depending on GeoSpark, such as Apache Sedona which used GeoSpark to activate spatial big data computing. We have decided to conducted this part of the research on GeoSpark to understand the performance rate of GeoSpark, and how much it can be worst according to the assumption in [12]. To illustrate this, the study was started by installing Apache Spark and open Spark localhost, then on Spark shell, Scala programming language was used and the dependency was added to activated GeoSpark on Apache Spark. However, the Spark build-in session or Spark SQL context was started,

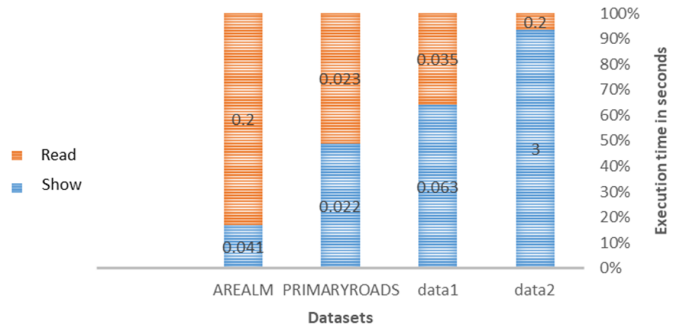

Fig. 3. A Compression among the Four Datasets when Read and Show their Contents on GeoSpark.

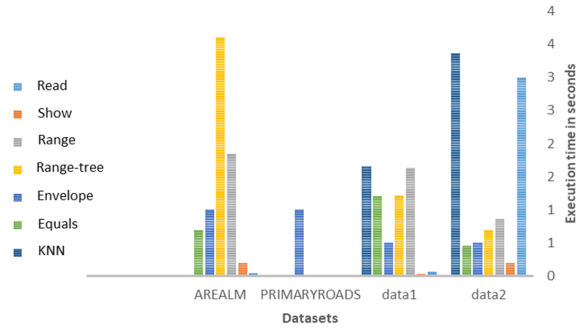

Fig. 4. Comparing the Execution Time for Different Functions and Queries on GeoSpark among the Four Datasets.

which is a class used for initializing the functionality of Spark SQL, where the Spark context is available as (sc), and Spark session is available as (spark). The datasets have been read using (spark.read), and execution time has been calculated for each dataset in the Apache Spark executor, and the comparison among the four datasets of the execution time of reading and show SQL functions in Fig. 3. However, as shown in Fig. 3 data 2 (8000 points geometry) has taken the highest reading time equals 3 seconds, where the show function for the same dataset takes only 0,2 seconds. On the other hand, the REALM dataset (129K polygon geometry) has recorded the lowest execution time for reading data on Spark equals 0.041 seconds. However, the ObjectRDD class was used for dealing with different spatial datasets geometries, such as PointRDD (data1 and data2), RectangleRDD (AREAL), and LineStringRDD (PRIMARYROA). However, two quires were examined with four operations on Geopark on the three datasets (data1, data2, AREALM), as shown in Table V. Additionally, PRIMARYROA datasets were presented multiple exceptions and errors so they excluded it and only datasets reading execution time was calculated. The Range query with tree indexing and without tree indexing has been applied on three datasets, KNN has applied only on the points datasets (data1 and data2), where Envelope and Equals have applied also on three datasets. After that, the execution time in seconds has estimated for each operation, and Table V shows more details about this step. Moreover, the obtained results were visualized in Fig. 4, which represented a comparison among the execution time required to examine the operations on the datasets. As a result, GeoSpark performance depends on the type of dataset geometries, but in general, the execution time of GeoSpark from our experimental study was acceptable.

IV. Results and Discussion

This section explained and discussed the main results that were observed from this research. However, the study was conducted on four main domains, and in each domain, was recorded some results, which were collected in this section. The observation was lead to that GeoSpark in Apache Spark supports better features than SpatialIgnite in Apache ignite, in which GeoSpark allow multiple file format, and multiple indexing, although that GeoSpark does not support spatial analysis as in SpatialIgnite. Moreover, SpatialIgnite supports more spatial functionality than GeoSpark, where KNN-query is only supported by GeoSpark. On the other hand, from experimental environment setup activation and working with spatial data computing on Apache Spark was more flexible than in Apache ignite, in which that preparing Apache Ignite experimental requirements have taken more time than Apache Spark, in which that spatial data computing on Apache Ignite required working on an IDE environment such as Netbeans, and adding some dependencies in the IDE environment, such as JTS Topology, was in Apache Spark does not depending on any IDE environment and the dependencies had been added on spark-shell in a simple code, and start working with the shell using Scala programming language. The last stage of the study built on the results and assumptions were produced by Md Mahbub Alam [12], which proposed that SpatialIgnite has addressed the best performance among GeoSpark, but from our observation, GeoSpark widely used for big spatial data processing, and many spatial frameworks have been depending on GeoSpark, such as Apache Sedona which used GeoSpark to activate spatial big data computing. Consequently, we have decided to conduct this part of the research on GeoSpark to understand the performance rate of GeoSpark. The result from this last step as it shows in Table V is mostly acceptable, but it depends on the size and the type of the dataset. However, the assumption by Md Mahbub Alam in his research can not be generalized on all Geospark and SpatialIgnite researches, in which multiple factors affected the performance of Geospark on Apache Spark, such as machine performance (CPU, and memory), dataset size, and type, programming environment (such as using spark-shell or IDE software). In the end, we have concluded that each one of these frameworks has its advantages over the other one, and each one of them can be used depending on the research area requirements and status.

V. Conclusion

Spatial computing is becoming increasingly relevant with the widespread use of mobile devices. The rise in scale and value of location data have led to the creation of a variety of specialized spatial data processing systems. In this research, we have conducted a comprehensive evaluation of big spatial data

TABLE V. COMPARING THE EXECUTION TIME FOR DIFFERENT FUNCTIONS AND QUERIES ON GEOSPARK AMONG THE FOUR DATASETS

Predicates	AREALM	PRIMARYROADS	data1	data2
Read	0.041	0.022	0.063	3
Show	0.2	0.023	0.035	0.2
Range	1.84	-	1.63	0.86
Range-tree	3.6	-	1.22	0.69
KNN	-	-	1.65	1.37
Envelope	1	-	0.5	0.5
Equals	0.70	-	1.2	0.46

computing on two data management systems Apache Ignite and Apache Spark. The comparative has been done on four different domains, experimental environment setup, supported features, supported functions and queries, and performance and execution time. The research concluded that each one of these frameworks has its advantages over the other one, and each one of them can be used depending on the research area requirements and status. However, the type and size of the dataset can have a large impact on the execution time. However, multiple factors affected the performance of Geospark on Apache spark, such as machine performance (CPU, and memory), dataset size and type, programming environment (such as using spark-shell or IDE software). From the observation GeoSpark has recorded more flexibility than SpatialIgnite on Apache ignite. Moreover, the spatial analytic features in SpatialIgnite on Apache ignite make it more suitable with spatial data analytic applications, such as analyzing spatial data, that extracted from social media. Additionally, GeoSpark in Apache Spark is more suitable with spatial data queries and indexing because it supports complex spatial queries, such as kNN queries, which is not supported in SpatialIgnite on Apache ignite. Moreover, GeoSpark is an agile spatial data processing framework that can meet the changes in the requirements, and fault tolerance.

REFERENCES

[1] Thulara N Hewage, Malka N Halgamuge, Ali Syed, and Gullu Ekici. Big data techniques of google, amazon, facebook and twitter. *Journal of Communications*, 13(2):94–100, 2018.

[2] Shadi A Noghabi, Kartik Paramasivam, Yi Pan, Navina Ramesh, Jon Bringhurst, Indranil Gupta, and Roy H Campbell. Samza: stateful scalable stream processing at linkedin. *Proceedings of the VLDB Endowment*, 10(12):1634–1645, 2017.

[3] Raghu Ramakrishnan, Baskar Sridharan, John R Douceur, Pavan Kasturi, Balaji Krishnamachari-Sampath, Karthick Krishnamoorthy, Peng Li, Mitica Manu, Spiro Michaylov, Rogério Ramos, et al. Azure data lake store: a hyperscale distributed file service for big data analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 51–63, 2017.

[4] Michael P Cope, Elena A Mikhailova, Christopher J Post, MA Schlautman, and P Carbajales-Dale. Developing and evaluating an esri story map as an educational tool. *Natural Sciences Education*, 47(1):1–9, 2018.

[5] Gouri Sankar Bhunia, Pravat Kumar Shit, and Debashish Sengupta. Free-open access geospatial data and tools for forest resources management. In *Spatial Modeling in Forest Resources Management*, pages 651–675. Springer, 2021.

[6] Malte Lech, Juha Ilari Uitto, Sven Harten, Geeta Batra, and Anupam Anand. Improving international development evaluation through geospatial data and analysis. *International Journal of Geospatial and Environmental Research*, 5(2):3, 2018.

[7] Salman Salloum, Ruslan Dautov, Xiaojun Chen, Patrick Xiaogang Peng, and Joshua Zhexue Huang. Big data analytics on apache spark. *International Journal of Data Science and Analytics*, 1(3-4):145–164, 2016.

[8] Michael A. Whitby, Rich Fecher, and Chris Bennight. Geowave: Utilizing distributed key-value stores for multidimensional data. In Michael Gertz, Matthias Renz, Xiaofang Zhou, Erik G. Hoel, Wei-Shinn Ku, Agnès Voisard, Chengyang Zhang, Haiquan Chen, Liang Tang, Yan Huang, Chang-Tien Lu, and Siva Ravada, editors, *Advances in Spatial and Temporal Databases - 15th International Symposium, SSTD 2017, Arlington, VA, USA, August 21-23, 2017, Proceedings*, volume 10411 of *Lecture Notes in Computer Science*, pages 105–122. Springer, 2017.

[9] Ahmed Eldawy, Louai Alarabi, and Mohamed F. Mokbel. Spatial partitioning techniques in spatial hadoop. *Proc. VLDB Endow.*, 8(12):1602–1605, 2015.

[10] Louai Alarabi, Mohamed F Mokbel, and Mashaal Musleh. St-hadoop: A mapreduce framework for spatio-temporal data. *GeoInformatica*, 22(4):785–813, 2018.

[11] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. Spatial data management in apache spark: the geospark perspective and beyond. *GeoInformatica*, 23(1):37–78, 2019.

[12] Md Mahbub Alam, Suprio Ray, and Virendra C Bhavsar. A performance study of big spatial data systems. In *Proceedings of the 7th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 1–9, 2018.

[13] Jia Yu, Zishan Fu, and Mohamed Sarwat. Dissecting geosparksim: a scalable microscopic road network traffic simulator in apache spark. *Distributed Parallel Databases*, 38(4):963–994, 2020.

[14] Louai Alarabi. Summit: a scalable system for massive trajectory data management. *ACM SIGSPATIAL Special*, 10(3):2–3, 2018.

[15] Yuanyuan Chen, Jingyi Yu, and Yong Gao. Detecting trajectory outliers based on spark. In *25th International Conference on Geoinformatics, Geoinformatics 2017, Buffalo, NY, USA, August 2-4, 2017*, pages 1–5. IEEE, 2017.

[16] Esteban Zimányi, Mahmoud Attia Sakr, and Arthur Lesuisse. Mobilitydb: A mobility database based on postgresql and postgis. *ACM Trans. Database Syst.*, 45(4):19:1–19:42, 2020.

[17] Aleem Akhtar. Role of apache software foundation in big data projects. *arXiv preprint arXiv:2005.02829*, 2020.

[18] Hadoop. Apache Hadoop. https://hadoop.apache.org/, 2021. [Online; accessed 23-Augst-2021].

[19] Spark. Apache Spark. https://spark.apache.org/, 2021. [Online; accessed 23-Augst-2021].

[20] Ignite. Apache Ignite. https://ignite.apache.org/, 2021. [Online; accessed 23-Augst-2021].

[21] Jia Yu, Jinxuan Wu, and Mohamed Sarwat. Geospark: A cluster computing framework for processing large-scale spatial data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–4, 2015.

[22] Andrey Tapekhin, Igor Bogomolov, and Oleg Velikanov. Analysis of consistency for in memory data grid apache ignite. In *2019 Ivannikov Memorial Workshop (IVMEM)*, pages 46–50. IEEE, 2019.

[23] Yassine Benlachmi, Abdelaziz El Yazidi, and Moulay Lahcen Hasnaoui. A comparative analysis of hadoop and spark frameworks using word count algorithm. *International Journal of Advanced Computer Science and Applications*, 12(4), 2021.

[24] Cristiana-Stefania Stan, Adrian-Eduard Pandelica, Vlad-Andrei Zamfir, Roxana-Gabriela Stan, and Catalin Negru. Apache spark and apache ignite performance analysis. In *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pages 726–733. IEEE, 2019.

[25] Zhibo Sun, Hong Zhang, Zixia Liu, Chen Xu, and Liqiang Wang. Migrating gis big data computing from hadoop to spark: an exemplary study using twitter. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 351–358. IEEE, 2016.

[26] Randall T Whitman, Michael B Park, Bryan G Marsh, and Erik G Hoel. Spatio-temporal join on apache spark. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–10, 2017.

[27] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. Spatial data management in apache spark: The geospark perspective and beyond. *Geoinformatica*, 23(1):37–78, 2019.

[28] Jayati Gandhi, Nekita Chavhan, and Girish Kumar. Comparative study of spatial hadoop and geospark for geospatial big data analysis. *system*, 9:10, 2020.

[29] Mingjie Tang, Yongyang Yu, Qutaibah M Malluhi, Mourad Ouzzani, and Walid G Aref. Locationspark: A distributed in-memory data management system for big spatial data. *Proceedings of the VLDB Endowment*, 9(13):1565–1568, 2016.

[30] Francisco García-García, Antonio Corral, Luis Iribarne, George Mavrommatis, and Michael Vassilakopoulos. A comparison of distributed spatial data management systems for processing distance join queries. In *European Conference on Advances in Databases and Information Systems*, pages 214–228. Springer, 2017.

[31] Zhou Huang, Yiran Chen, Lin Wan, and Xia Peng. Geospark sql: An effective framework enabling spatial queries on spark. *ISPRS International Journal of Geo-Information*, 6(9):285, 2017.

[32] Panagiotis Nikitopoulos, Akrivi Vlachou, Christos Doulkeridis, and George A Vouros. Distrdf: Distributed spatio-temporal rdf queries on spark. In *EDBT/ICDT Workshops*, pages 125–132, 2018.

[33] Suprio Ray, Bogdan Simion, and Angela Demke Brown. Jackpine: A benchmark to evaluate spatial database performance. In *2011 IEEE 27th International Conference on Data Engineering*, pages 1139–1150. IEEE, 2011.