Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 2, February 2018, pp. 320-333
DOI: 10.23940/ijpe.18.02.p13.320333

A Survey on Trajectory Big Data Processing

Amina Belhassena*, Hongzhi Wang

Harbin Institute of Technology, Harbin, 15007, China

Abstract

Rapid advancements of location-based information provided by publicly available GPS-enabled mobiles devices boost the generation of massive trajectory data. Recently, numerous researchers have addressed many problems regarding trajectory data, which is based on storage and queries processing. Further, a wide spectrum of application domains can benefit from trajectory data mining including trajectory organization as well as queries. Therefore, large-scale trajectory data has received increasing attention in research fields as well as in industry. As the massive trajectory data processing exceeds the power of centralized approaches used previously, in this paper, we survey various existing tools used to process large-scale trajectory data in a distributed way, e.g. MapReduce, Hadoop, and Spark. Furthermore, this paper reviews an extensive collection of existing applications of movement objects, including trajectory data mining and frequent trajectory. We also propose an open interesting research direction, which is challenging and has not been explored up until now, to improve the quality of trajectory query.

Keywords: trajectory data; query processing; distributed platforms

(Submitted on November 21, 2017; Revised on December 18, 2017; Accepted on January 15, 2018)

1. Introduction

Data needs to be organized, processed, and gathered from massive datasets referred to as big data, which is a blanket term for technologies and non-traditional methods. Nowadays, such data exceeds the computing capacity of conventional databases as well as the storage of a single computer; therefore, the value, scale and pervasiveness of such data computing and analysis has been widely developed recently.

Since the data is too large, moves too fast, and does not fit the strictures of database architectures, these data have emerged from the convergence of rapidly decreasing costs for collecting, storing, processing, and then sharing data. Many application domains have developed efficient approaches to process and analyze big data such as healthcare, manufacturing, personal location data, finance in the public sector, etc.

Nowadays, Geographic Information System (GIS) is used to help people in their daily lives, for example, finding restaurants, hotels, or bus stations, or providing accurate traffic information. Many companies play an important role in business development, and by using the power of big data, these companies are now able to maximize the capacity of GIS to drive business growth. Using business technology devices like smartphones or tablets can attract the attention of and provided useful information to users.

With the proliferation of data, the geographic information system uses some devices like GPS (Global Positioning System) and sensor technologies to collect positional data. These large amounts of data capture the motion history of moving objects, called trajectories. A trajectory is an ordered series of locations, where each location is called a point of interest and has an associated description such as: latitude, longitude, name, and other text description like activities. The big trajectory data processing field is meaningful in several aspects.

* Corresponding author.
E-mail address: amina_belhasna@hotmail.fr

Firstly, such moving objects are archived in TraJectory DataBases (TJDBs) [3] for processing and deep analysis to discover the knowledge and support decision-making.

For example, in trajectory marine tracking domain, the automatic identification system (AIS) installed by vessels is used to analyze the massive data in order to detect abnormal activity in the routes of vessels. On the other hand, the application maps installed in smartphones cooperate with business in public transportation to collect and store the trajectories of the history users in TJDBs for analyses aiming to develop and improve the quality of their services.

Secondly, the trajectories extracted from TJDBs and their queries are not banal.

Thus, in the last two decades, in order to process and analyze the trajectory data from TJDBs, many similarity methods have been widely studied. The trajectory similarity search has several trajectory query forms, and among them is a keyword based trajectory search. This kind of query is important to efficiently handle trajectory data challenges. For example, the visitor in a new city may want to know the minimal trajectories from TJDBs similar to its query. The query consists of a combination of associated geo locations with keyword information. It aims to find the K trajectories that contain the most relevant keywords based on minimal distance.

Thirdly, recently, the TJDB contains the trajectories formed from large-scale data of sequence objects including their text description, resulting in a large number of sub-trajectories as well as long trajectories to check. Hence, the process of such data needs more computation and exceeds the power of single machine used previously. Thus, efficient traversing of the TJDB in order to extract the knowledge and discover trajectories has become an important task in the research field as well as in industry.

Due to this importance, trajectory data processing draws attention in the research field as well as in industry. Hence, many distributed methods have been proposed. For the convenience of the survey, we classify existing platforms. The major similarity between classes are based on the MapReduce model processed in distributed platforms. Moreover, the major dissimilarity between classes are time consuming as well as reuse data cost. The trajectory data can be processed through the following platforms:

- Hadoop framework.
- Apache Spark.
- Other platforms.

As stated above, trajectory big data processing attracts more and more attention to improve user satisfaction, therefore it is essential to conclude and summarize the existing works in related literature. This survey performs just this task. It firstly presents the multiple classification platforms used to process the big trajectory data and some research results in specialized search areas, together with some novel techniques. Then, it presents some trajectory query techniques that are used in specialized fields based on distributed methods. In this study, our contributions are summarized as follows:

- We initially and briefly present the MapReduce model as well as the different studies proposed to handle large-scale spatial data.
- As MapReduce is not sufficient to handle massive spatial data, especially trajectory data, we present the different effective platforms, their advantages and their drawbacks.
- Since spatial query is important to handle large-scale spatial data, thus, we present different search results and query approaches to efficiently handle the distributed query challenges based on numerous platforms.
- Finally, we summarize the comparison between the different distributed systems, the limitation of the previous works related to trajectory query, and then we propose a useful solution through this survey to handle the mining trajectory data query challenge.

2. Trajectory big data processing

The increasing expansion of mobiles devices, sensors and numerous GPS tools has led to an explosive increase of the movement of spatial objects producing the trajectories. Nowadays, large-scale trajectory data is widely available. However, its management, processing and use become difficult due to its big volume, large data variety and quality, and the challenge of the meaning of trajectories is one of the crucial domains for big data analytics. In this section, we will discuss recent research works in trajectory data management, performance processing and limitations. We will present some popular tools used recently to process trajectory big data.

To deal with the huge amount of trajectory data and its complexity, the distributed data processing system can be used to conquer this challenge. During the last decade, there has been an interesting development in parallel and distributed computing fields applied to the management, analysis and processing of massive geospatial data. Many applications are used in real life, such as smart cities, weather analysis, traffic monitoring, etc., which yield a great performance and a fast response time that cannot be met by classical processing methods of geographic information science (GIS) and spatial databases.

2.1. Spatial data performance based on MapReduce

The input spatial data are usually large and exceed the power of centralized methods. Thus, the parallel computation should be processed through hundreds or thousands of machines achieving a reasonable time during the operation. Therefore, the parallelism computation poses a major challenge in which data should be managed and distributed.

Also, failures conspired should be handled to not discover the original sampling computation with big amounts of complex code. Thus, MapReduce [7] is a programming model for processing and generating large datasets. MapReduce is based on two operations, which are map and reduce. These two operations permit programmers to parallelize a large amount of data in a straightforward manner. In addition, they can use re-execution as the primary mechanism for fault tolerance. Initially, the map operation is applied to each logical input record in order to compute a set of intermediate key/value pairs. Then, the reduce operation is applied to all values that share the same key in order to collect the derived data appropriately.

An example of map and reduce operations is from the GOOGLE Research Team [46].

- Input: large text documents
- Output: compute word count through all the document.
- Map: return map(word,1) for every word in document, where word is a key, 1 is a value.
- Reduce: sum all occurrence of words and return reduce (word, total-count).

Many different performance strategies and methods have been presented to process and analyze the big spatial data. These methods are implemented using MapReduce as a programming model in parallel computing systems. These systems are based on clusters and cloud computing [1] used for running large-scale spatial queries. In addition, the personal computers composed with chip multiprocessor CPUs and parallel GPUs [44] have achieved good support for data management and a low initial and operational cost.

[21] proposed and implemented a standard clustering algorithm dFoF for astronomical data. This algorithm was implemented on shared-nothing parallel data processing framework Dryad [19]. It used to analyze astrophysical simulation output using the MapReduce-style data analysis platform. In addition, it used scalable and parallel clustering for N-body simulation, resulting in the concerned domain. In order to meet the rapidly growing methods of Google's data processing needs, [14] have implemented and designed the Google File System GIS (GFS). GFS achieves scalability, reliability, availability, and fault-tolerance with a good performance.

2.1.1. Spatial complex query based on MapReduce

In the aim of processing spatial complex queries, MapReduce used to parallelize the massive data in order to process such kind of queries. [20] developed a method to process All Nearest Neighbors (ANN) queries. They started to adopt a splitting method for balancing workload based on a Z-curve processed in MapReduce. Then, they created a pending file structure and a redundant partition method in order to deal with relations between spatial object. After this, they proposed a strip-based two-direction sweep algorithm in order to accelerate the computation.

For spatial join queries, in order to directly support the heterogeneous related data sets processing problem, [18] proposes a new model called MapReduce-Merge. It aimed to add to MapReduce a Merge phase that can efficiently merge data already partitioned and sorted by map and reduce operations. They also proved that MapReduceMerge could express relational algebra operators as well as implement several join algorithms. [48] introduces the Spatial Join with MapReduce (SJMR), which is a parallel algorithm used to relieve the challenge. The SJMR algorithm used the strategy in which, at Map stage, SJMR partitioned the input data into disjoint partitions using a spatial partitioning function. Every partition was merged with a strip-based plane sweeping algorithm, tile-based spatial partitioning function and the technology of duplication avoidance.

2.1.2. Trajectory complex query based on MapReduce

Regarding the moving spatial object domain, efficient query processing and data analysis of such big trajectory data remains a big challenge. To deal with this problem, the first approach is introduced in [30], in which authors present a new framework based on MapReduce called PRADASE (Query Processing of Massive TRAjectory DAta baSEd on MapReduce). PRADASE relies on a GFS-style storage only supporting the appending data. They studied the method types appropriated for large cluster, and a set of them was improved to become a new partition strategy in order to save the continuity of trajectory data and distributed environment storage. Moreover, they propose the maintenance of all new updates in memory at each data node until the accumulation of particular sizes of data. Thus, the data writing in disk can largely improve the data update performance. Furthermore, to efficiently optimize the query processing over clusters, they propose two scalable indexing methods, which are PMI (Partition based Multilevel Index) and OII (Object Inverted Index).

On the other hand, [41] proposes TRUSTER (TRajectory Data Processing on ClUSTERs). They introduce a distributed indexing method, which supports query processing over massive trajectory data through clusters. TRUSTER divides the entire spatial space into statics partitions, where trajectories are distributed into theses partitions. Then, an index is constructed over temporal dimension for the trajectories in each partition. For the performance and the efficiency of the index creation and the queries, the nodes in the cluster could take charge of different spatial partition.

2.2. Trajectory complex query based on MapReduce

In this section, we present the different distributed platforms used for trajectory big data processing, the recent research results obtained using these platforms, and the process limitations occurring through these platforms.

2.2.1. Hadoop framework

MapReduce has recently become popular in programming models for processing and analyzing big spatial and no spatial data including the trajectory data. It is widely used in different applications of industry and theory methods of academia. Thus, several frameworks have adopted it, including the Hadoop framework.

Apache Hadoop is an open source framework developed in Java based on MapReduce, and it is used for distributed storage and processing massive data. Hadoop is presented as computer clusters built from commodity hardware, which offer various tools, such as Distributed File System (DFS), job scheduling and resource management capabilities. The core of Hadoop has an important component, the Hadoop Distributed File System (HDFS). For data processing, Hadoop initially splits data files into large blocs. Then, it distributes them through nodes in the cluster. After, it transfers the packaged code into nodes in parallel to process data. This method aims to access the manipulated data locality, which offers fast and efficient processing. The success of Hadoop has interested many researches and has led to many optimizations of and extensions to the framework.

For managing and analyzing massive data, the system chosen should provide a high performance and scale over clusters of thousands of machines, and it should be versatile [5]. The term versatility refers to the capacity of the system to analyze the complex queries. In addition, it refers to the flexibility of the system to support query languages, and controls the data preparation, location and management. Therefore, many research works have focused on evaluating the performance and scalability of data analysis technologies such as Hadoop based on MapReduce implementation and parallel data base system.

Several interface-level hybrids have developed recently such as Hive [36], Pig [31] and HadoopDB [4,5]. Hive is an open source data warehousing solution built on top of Hadoop supporting queries expressed in a SQL-like declarative language HiveQL. These queries were compiled into MapReduce jobs executed through Hadoop cluster. The Pig Latin is a language designed to fit in a sweet spot between the declarative style of SQL and the low-level, procedural style of MapReduce. It was fully implemented and compiled into physical plans that were executed over Hadoop. HadoopDB is an open source project. It is a hybrid of MapReduce and DBMS technologies, which combines the scalability features of Hadoop with the high performance of database systems for structured data analysis. It was designed to meet the growing demand of analyzing big dataset over large clusters.

2.2.2. Apache Spark Framework

Nowadays, instead of Hadoop, Apache Spark is well adopted in many real applications and research including trajectory domain. It was developed at UC Berkeley and several companies. Spark is fast and a general engine for large-scale big data, and it provides a convenient language-integrated programming interface similar to DryadLINQ [47,49]. With Spark, the applications can be written quickly using Java, Scala, Python or R. In addition, Spark can be used interactively to query big datasets from an interpreter like Scala. Furthermore, Spark has an advanced DAG execution engine, which supports acyclic data flow and in-memory computing. This feature proves that Spark runs programs up to 100x faster than Hadoop MapReduce in memory and 10x faster on disk. Spark runs on standalone cluster mode, EC2, Hadoop YARN, and Apache Mesos.

The data can be accessed through HDFS, Cassandra, HBase, Hive, Tachyon, and any Hadoop data source.

Spark uses Resilient Distributed Datasets (RDD) to abstract data that is to be processed. RDD is a collection of objects grown through a cluster, which can be stored in the RAM or on a disk. RDD offers a simple and efficient programming model for wide kinds of applications, and it provides rapidity when data fits into the collection of memory. RDD can be built across parallel transformation. The important feature of RDD is fault tolerance, where RDD can be automatically rebuilt on failure. It provides coarse-grained operations and tracking lineage.

2.3. Others Distributed Spatial Data Processing Platforms

Hadoop-GIS [1] is a scalable and high performance spatial data warehousing system over MapReduce. It has been developed to support large-scale spatial queries through Hadoop cluster. Hadoop-GIS achieves a task parallelization by providing data partitioning. In order to process various spatial query types, Hadoop-GIS provides an indexing method for spatial query, an implicit query parallelization strategy across MapReduce. To generate the correct results, Hadoop-GIS furnishes an effective method by handling boundary objects. In order to support the declarative spatial queries, Hadoop-GIS is efficiently integrated with Hive, providing an expressive spatial query language. Hadoop-GIS supports basic spatial queries such as point, containment, join, and complex spatial queries such as spatial cross-matching which are the spatial join over large-scale and nearest neighbor queries. In addition, the structured feature queries are integrated with spatial queries and supported through Hive.

Spatial Hadoop [9] is a comprehensive extension based on MapReduce to Apache Hadoop. It has been designed to handle and analyze spatial data through the cluster of several machines. It achieves a better performance compared to Hadoop when dealing with spatial data. It employs a simple high-level language and two-level spatial index structures, which can be built in HDFS such as R-Tree, R+-Tree and Grid file. In addition, it employs the basic spatial components built inside the MapReduce layer, and three basic spatial operations: range queries, k-NN queries, and spatial join. The Spatial Hadoop permits users to implement custom spatial operations supporting the spatial indexes. Moreover, in order to make querying intuitive and easier, Spatial Hadoop comes regrouped with Pigeon, which is a spatial extension to Pig [31] based on ESRI Geometry API. Pigeon integrates a function that facilitates the use of the existing PostGIS to users.

For distributed on-line queries platforms, a distributed key-value store, Apache Cassandra [22], has been developed to handle massive amounts of data spread out through many commodity servers. Cassandra is developed at Facebook. In order to provide the availability without single point failure, Cassandra uses data replication across multiple data centers. In addition, for data updating, Cassandra give us the opportunity to choose the replication types, which are either synchronous or asynchronous.

On the other hand, a distributed open source called HBase consists of a column-oriented database system based on Google BigTable [6]. HBase runs on Hadoop and Zookeeper [17] and stores data into HDFS files. HBase provides scalability, replication data and fault tolerance. Based on HBase's API, HBase can be accessed for real time access using MapReduce tasks in Hadoop batch processing analytics. HBase resembles Cassandra because both are data models that provide the structure of key-value stores. This means different keys can have different columns. HBase is designed with high performance to intensively read workloads, whereas Cassandra is a write-oriented system.

To implement the efficient functionality of HBase, Apache Phoenix, which is a SQL layer, uses HBase API through co-processors and custom files. For SQL query processing, Apache Phoenix compiles it into a series of HBase scans, and

organizes the results to deliver a set of regular JDBC results to the clients. The meta-data is stored by Phoenix through HBase.

Furthermore, as Hive is not suitable for online queries, Stinger has been developed to improve Hive [36]. Stinger used to build interactive querying support over Hive. The performance of Stinger has been improved up to 100x.

In order to handle to management and analysis of large scale trajectory data, SECONDO [12] is designed as an extensible database system. It provides a large number of data types and algorithms to represent and process the massive trajectory data efficiently.

3. Trajectory query processing based on distributed platforms

In order to support different spatial query types, most of the MapReduce-based systems presented above either lack handling of spatial query processing or do not have the ability to support the spatial query. MapReduce aims to fit nicely with large-scale problems across key-value based partitioning, but due to the multi-dimensional nature of spatial queries and analytics [2], they are naturally complex and difficult to fit into MapReduce model. In order to handle spatial data partitioning, there are two major problems which should be take into account. The first problem consists of load imbalance of tasks in distributed systems and high cost of response time. The second problem consists of an inability to handle the query properly, which yields incorrect query results. As the spatial query is naturally complex in order to efficiently prune the search space and requires a high cost of geometric computation, efficient access methods are thus needed. To handle spatial complex queries as well as trajectory queries, an effective distributed framework should be necessary. We present in the following parts some recent research works that have adopted major famous distributed approaches based on both the Hadoop and Spark frameworks.

On the other hand, since the ordered sequence of massive moving objects are archived in TJDBs, the routes on which these objects are connected are also stored in TJDBs. Therefore, the statistics related to the frequent trajectory is integrated in the massive TJDBs. Thus, such knowledge is necessary in many application domains. Thus, we present recent works that have adopted the frequent trajectory challenges.

3.1. Spatial Query based On Hadoop System

As Hadoop recently become famous regarding its trajectory data processing domain, we present some recent research focusing on efficient trajectory query processing and massive trajectory data analysis over Hadoop based on the MapReduce model.

In order to help users to convert large-scale sequential queries to parallel queries in an efficient manner without having to learn the MapReduce programming model, [23] proposed a coupling structure, on which they combined a single computer database with Hadoop on the engine level. This structure aimed to implement the current and future data types and algorithms directly, without changing anything special for parallel platform. In addition, the data was transferred through an independent distributed file system into the database directly, without using HDFS, reducing the unnecessary transform and transfer overheads. To achieve an impressive speedup on all steps of large-scale queries, authors used the parallel SECONDO in their demonstration. [24] proved the efficiency of parallel SECONDO by using it to handle massive trajectory data of urban application. Initially, to build up the urban traffic, they imported the data from OpenStreetMap. After that, they started to process the network-based queries like map-matching and symbolic trajectory matching. The implicated queries were fixed as sequential expressions with a consumed time in single-computer SECONDO. Furthermore, using the parallel SECONDO, authors demonstrated that the queries can achieve an impressive performance, obviously after the conversion of the sequential queries to parallel queries. In addition, the demonstration was just done on a small cluster of six low-end computer machines.

For marine tracking purposes, authors in [39] presented an approach that consisted of two levels to detect abnormal activities in the routes of vessels. Vessels are required to install a self-reporting system, which is an Automatic Identification System (AIS) with transceivers and transponders. The data is crewed from AIS. Initially, for pre-clustering data points, they developed a clustering algorithm Density-based Spatial Clustering of Applications with Noise considering Speed and Direction (DBSCANSD). To know whether the data types are normal or abnormal, they got the results of the labeled points based on the help of the experts of domain knowledge in maritime. In the second level, they used the labeled data generated

in the first level to train the Parallel Meta-Learning (PML) algorithm on Hadoop based on MapReduce. They proved that PML provides the scalability, and both accuracy and time complexity results were improved when increasing the node number in the cluster.

Continuous with emerging smart city strategies, especially urban data challenges such as processing of floating car data and transportation data [16,43], authors in [16] addressed the problem of transportation data processing. This data was built in the transportation data center to simultaneously collect and pre-process a massive traffic volume using GPS devices, and share them to various public services and modern integrated transportation managements. In order to estimate the correct GPS location points on corresponding road links, Map Matching (MM) can be used to address the problem. With massive GPS data, authors in [16] proposed an improved parallel topological map-matching algorithm. The parallelization design solved once challenged in serial MM, such as slow data sorting, large search space, and long map loading time. In addition, in order to gain more power from connected computers, they used MapReduce model to parallelize the proposed algorithm over Hadoop. The algorithm and the method proposed achieved highest efficiency as well as guaranteed accuracy with a reduced time. Moreover, as is mentioned in the literature, the urban computing and smart transportation systems may use data gathered from GPS records. The collection of these records produces the floating car data. Many problems can be handled using the floating data such as traffic congestion and travel times. Furthermore, floating data can design the characteristics of urban crowd flow using the data collected from taxi trajectories across the city. However, with the massive floating data car, it needs an efficient method for processing and analyzing. Thus, in [43], authors used a Hadoop-GIS (which is presented the above section) methodology. It consisted of two major tools provided by ESRI's, which are the spatial framework for Hadoop and geo-processing tools for Hadoop. They adopted more than 1 billion floating car data in Beijing over 17 days in November 2014, which was generated from about 32,000 GPS-equipped taxicabs. The data extracted from Hadoop-GIS based on MapReduce was used to discover the crowed flow pattern in Beijing in a way to provide suggestions for city vehicle managements. After analysis, they proved that Hadoop-GIS could process floating car data efficiently and effectively. The model employed in the paper could attempt to process in parallel large scale floating car data sets through Hadoop-GIS, effectively providing an efficient data storage.

3.2. Spatial Query Based on Spark System

Recently, with the efficiency of Spark framework, some researchers have adopted it to process the spatial and no-spatial query, analysis, and management of spatial massive data in parallel. Authors is [42] developed a framework in-memory computing cluster, which is called GeoSpark. GeoSpark handled spatial data, and it provided an API for Apache Spark programmers to develop spatial analysis applications in a straightforward manner. It supported spatial access methods such as R-Tree using Spark framework. It consisted of three layers, which were Apache Spark Layer, Spatial RDD Layer and Spatial Query Processing Layer. GeoSpark improved the performance of previous works, which were based on the MapReduce model, and it achieved a better run time performance than that of Spatial Hadoop.

On the other hand, authors in [8] designed a new system called Spatial In-Memory Big data Analytics (SIMBA). It is a distributed in-memory analytics engine, which is used to support spatial queries and analytics over massive amounts of spatial data. SIMBA aimed to provide a simple and expressive interface for programmers. In addition, it achieved scalability, a low query latency, and high analytics throughput. SIMBA is based on Spark and runs through a cluster of computer machines. In order to support spatial queries and analytics, SIMBA extended the Spark SQL with a class of spatial operations providing a simple interface using both SQL and DataFrame API. Furthermore, in order to achieve a low query latency, SIMBA introduced indexes over RDDs (resilient distributed dataset). Moreover, to support multiple parallel spatial queries, SIMBA had a SQL context module improving the analytical throughput. In addition, to achieve and optimize both low latency and high throughput, SIMBA implemented an effective query optimizer, and to select good spatial query plans, it used cost-based optimizations (CBO). This system is based on Spark, and thus, it extended the fault tolerance provided by Spark, and an excellent scalability was achieved when spatial queries and analytics were answered over big spatial data.

Regarding trajectory spatial data domain, [26] addressed the query processing problem of trajectory aggregate queries. They used a spatio-temporal index and developed an inverted index to trajectory data in order to the design random index sampling (RIS) algorithm, which estimated answers with a guaranteed error bound. Furthermore, in order to improve the scalability of the system, they extended the RIS algorithm to the concurrent random index sampling (CRIS) algorithm, aiming to process a number of trajectory aggregate queries coming simultaneously with overlapping spatio-temporal query regions. Their system was run over a cluster of machines, and the query system is built on top of Spark in-memory based on MapReduce model.

For trajectory stream, which consists of the sequence of spatial locations of a moving object over time, each point has its own definition including at least a trajectory ID, latitude, longitude, and a timestamp. In order to address the technical challenge of implementing the parallel trajectory stream analysis, authors in [40] defined the trajectory analysis issue as discovering trajectory companies of moving objects. They proposed an efficient technique respecting the data partition strategy, workflow design, and optimization of workflow steps. In order to avoid the unbalanced workload and data screw, they efficiently distributed data over the cluster. Furthermore, to handle the data-shuffling problem over the network, they developed an efficient workflow in parallel. In addition, based on Spark cluster-in memory, they proposed optimization methods to improve the performance.

3.3. Mining trajectory big data processing

Efficient data processing for mining massive trajectory data research has become an emerging and challenging task in real life application as well as in industry. MapReduce based on Hadoop cluster is still the major solution to handle such types of data challenges. Many works have addressed to handle the mining trajectory data such as [11,15,35,38,49].

In order to process and analyze the massive trajectory data based on MapReduce through Hadoop, authors in [35] developed an efficient algorithm in the way to investigate the problem of spatial join between a big set of spatio-temporal trajectory data and a set of spatial regions POIs. They presented a trajectory as the movement of a large collection of moving objects/mobile users. These collections were racked for a certain period, characterized by a specified frequency of location updates. As trajectory data volume is large, each spatial region represents an area around POI visited by users who stayed there for a specific period. Thus, they divided the processing into two cases. In the first case, the recording location of the user at the corresponding time stamp was within the area of POI, meaning the user visited a particular POI. Otherwise, the user was considered to be on a trajectory between two POIs. In such analysis, using the MapReduce model, they developed two MapReduce jobs over trajectory and the POI region. The first job is called SemanticTrajectory, which performed the process and analysis of trajectory data related to users; it also detected their visits to POIs. The second job is called PopularPlaces. It aims to focus on POIs and detects their popularity according to the number of visits and the total duration of stays. The proposed approach proved the feasibility of MapReduce and Hadoop usage in big spatio-temporal trajectory data for processing and analyzing.

Recently, the discovery of the trajectory data-mining pattern of restricted moving objects has become a great challenge in urban services. It usually faces common problems such as data storage and processing intensity, time demand, and coordinate system transformation [49]. Thus, authors in [49] addressed these challenges by presenting an efficient framework consisting of three modules, which are: data distribution module, data transformation module, and I/O performance improvement module. Initially, in order to distribute data, they proposed a two-stage consistent hashing algorithm. It distributed data to multiple nodes in the cluster, and the data of each node was also distributed to multiple disks. For the load balancing strategy, they optimized it by adding a load-adjusting step. Furthermore, in order to transfer data efficiently, they proposed a parallel linear referencing strategy based on the MPI parallel programming framework. It achieved the performance for interaction overhead reducing and the coupled interactions reducing. Moreover, to improve I/O performance, they designed a compression aware method. The proposed method analyzed the I/O performance of data compression and decided whether data can be compressed. In addition, it intelligently achieved the balance compression rate and ratio.

In the traffic subarea division, the parallel processing paradigm MapReduce based on Hadoop cluster handles the scalability of massive trajectory data. It provides efficient computing framework to deal with big taxi trajectory data [38]. Authors in [38] developed an efficient parallel K-mean algorithm based on MapReduce (Par3PKM) through Hadoop to solve the division problem of traffic subarea. The time-consuming iterations of the proposed algorithm were performed in three phases, which were the Map function, the Combiner function, and the Reduce function. Since the process of the incremental Combiner function was done between the map tasks and Reduce tasks, the computational complexity of MapReduce jobs was reduced, and the limited bandwidth available was stored on a Hadoop cluster.

Furthermore, the K-mean algorithm with its optimization was also used to pre-process the massive moving data captured from vehicles equipped by GPS devices. In addition, it was used to analyze the rapid growth of this data. As Hadoop and MapReduce became sophisticated tools in order to overcome big data challenges in industries, including transportation domain. Thus, with such size, authors in [14] divided the large number of data into partitions to facilitate handling an unstructured database. They used the K-means algorithm with its sequential mode and the parallel mode based on

MapReduce over Hadoop. The proposed parallel k-means algorithm was tested with their massive GPS data, achieving a fast computation and efficient results.

The trajectory data that consists of time and spatial dimensions, named as spatiotemporal data, has greatly enhance the content of data itself. This data is characterized by its huge amount, sequential mode, spatial type, and dynamic feature, etc. In order to process the massive spatio-temporal trajectory data efficiently, computing could be an efficient solution providing a large-scale data, efficient computing model with high-quality storage resources, computing resources, and cyber sources [16]. In order to address this spatio-temporal trajectory data challenge, authors in [16] developed an efficient fast parallel trajectory clustering method for big data on MapReduce based on cloud computing platform. Initially, they presented a fast calculation algorithm based on coarse-grained Dynamic Time Warping, aiming to solve the similarity between time series with different sampling rates. The similarity algorithm proposed efficiently reduced the time cost when the trajectory length was very long. After that, they proposed the parallel trajectory clustering strategy of trajectory big data under the Hadoop MapReduce model. The strategy proposed was based on K-mean algorithm was processed in three phases, which were Map phase, Shuffle phase, and reduce phase. They proved that the parallel trajectory clustering was valid. Moreover, compared with the traditional clustering algorithm, when the data size increased, the parallel method proposed achieved the performance efficiently and effectively.

As we mentioned before, MapReduce and Hadoop were adopted in many big trajectory data applications. This is due to the Hadoop scalability's features, which provides good characteristics for data access. It also provides an efficient fault tolerant. Using HDFS files through Hadoop, the massive data could be rapidly accessed, as HDFS is designed for sequential read data. For example, if data is composed of five blocks and the data located in the fourth block needs to be read, the process in HDFS recommences from the first block and reads all the blocks in sequences. Therefore, HDFS is highly optimized for parallel sequential data reading, providing a lower cost time and a speedup processing [38]. However, with the fast growth of data type including trajectory data set, there are a number of Hadoop and MapReduce limitations, which weaken the management, and the processing of huge numbers of data.

To ensure a full data recovery during processing in case something goes wrong, MapReduce writes all of the data back to the physical storage medium after each operation, which occurs a high cost of I/O disks because data that is held electronically in RAM is more volatile than that stored magnetically on disks. Furthermore, using HDFS files MapReduce reuses a set of data across multiple parallel operations. It needs to reload data from the HDFS files stored in the disk in each job and in each iteration, thus requiring a big amount of time writing and reading. This makes it inefficient for an important class of emerging applications. Data reuse is common in many iterative algorithms such as machine learning and graph, PageRank, K-means clustering, and logistic regression [47].

3.4. Mining frequent trajectory big data processing

Discovering frequent pattern mining was first proposed by [33]. It aims to find interesting patterns from a large database such as association rules, correlations, sequences, episodes, classifiers and clusters. The association rule mining remains to be an effective method to extract the knowledge information from the query and large datasets. The association rule mining is an important method used to extract the knowledge information from the query and a large dataset. Its purpose consists of finding a repeated probability p between the values of two or more data in the transaction dataset items. If p is high, that means an association exists between data items, which can be established among them.

Many works introduced the sequential pattern mining based on frequent patterns and association rules mining in trajectories. [28] and [29] presented a method to extract the association rules from moving object databases then returned the best trajectory using a matching function, where [28] implemented a modified version of the Apriori algorithm and [29] used a modified version of the PrefixSpan algorithm. Moreover, [10] and [27] aimed to find the frequent subsequences in trajectories. [10] introduced an approach based on partitioning strategy to reduce the trajectory size and present the trajectory as strings. They then utilized the windows approach that was mixed with a counting algorithm to mine the frequent trajectories. [27] presented a new technique to predict the next location of a moving object. They built a decision tree named T-pattern tree to hold a certain area, which might be used to find the best path of the new trajectories.

However, data is increasing in different fields, where its amount will be doubled every two years according to the statistics released by the authority in 2011 [13], and in 2020 it is expected that the data measure of human beings will

achieve a staggering 35 trillion GB [13]. Therefore, all the methods cited above that implemented the frequent data mining algorithms in a centralized platform have become insufficient with tending to the big data usage nowadays.

Some researchers used parallel computing to implement the Apriori algorithm in distributed platform such as [13,25], and [45] used the MapReduce model. Due to the large database, in order to efficiently develop fast algorithms that can handle massive data mining processing, the frequent mining Apriori algorithm is one of the famous algorithms. It used to handle such kinds of tasks in parallel based on MapReduce paradigm [25]. In order to efficiently develop an application for online bookstore recommendations, authors in [13] developed an efficient algorithm called CMR-Apriori based on coding and MapReduce model through Hadoop. The algorithm aimed to identify the corresponding rules in the knowledge model with a quick and accurate manner. They improved CMR-Apriori algorithm by utilizing the massive amount of data on a distributed HBase, which presented the input of MapReduce.

However, as the frequent itemset mining process is both data-intensive and computing intensive. That means, large scale dataset challenges are handled in data mining. In addition, the frequent mining algorithm needs to iteratively scan the dataset many times, which generates valid information. Thus, the frequent itemset mining consumes a large time over massive data. Therefore, the distributed computing and parallel methods have become effective for accelerating the computation of large-scale dataset algorithms. However, the existing methods cited above that implemented the parallel Apriori algorithm based on MapReduce are not efficient enough for iterative computation. [32] presented a Yet Another Frequent Itemset Mining algorithm (YAFIM), which is a parallel Apriori algorithm based on Spark in-memory RDD framework. It designed in parallel computing to support iterative algorithms and iterative data mining. Initially, with YAFIM, the transactional data was loaded from HDFS file into Spark RDD. Then, k-frequent itemset was used to iteratively generate (k+1)-frequent itemsets. YAFIM achieved a good speedup, where it outperformed the existing method based on MapReduce with a speedup of 18 x.

Moreover, YAFIM aimed to generate candidate sets that had all possible pairs for singleton frequent items and compared each pair with every transaction record. Thus, the computation yielded expensive tasks. Therefore, [34] improved YAFIM by presenting a new approach called Reduced-Apriori (R-Apriori), which dramatically reduced the computational complexity based on SPARK framework in-memory. The method eliminated the candidate generation step and avoid costly comparisons.

4. Discussion

In this section, we propose a summary of the systems used to process spatial trajectory data, discuss their limitations, and propose the effectiveness of certain systems that could handle large-scale trajectory data.

4.1. Classification summary

Initially, we compare the major systems, which are Hadoop and Spark frameworks. Table 1 presents the comparison based on some parameters. The main advantage of Spark compared to Hadoop is the data storage type. While processing, Spark stores all data in-memory, on which it achieves a lower cost compared to Hadoop, which uses a hard disk to store all the required data for processing. Hence, due to memory computation, Spark is faster than Hadoop.

Moreover, the spatial query that should be adopted can be mapped into partition based query processing framework, preserving the correct query semantics. Therefore, some researchers have adopted, developed and redesigned interesting systems to take into consideration the spatial query methods based on MapReduce computing infrastructure. Table 2 summarizes the major advantages and disadvantages of some platforms based on MapReduce. HadoopGis is characterized by the following limitations:

- HadoopGis processes spatial data as non-spatial data, without any use of other additional support.
- It supports only uniform grid, and it can be applicable only in uniform data distribution.
- With HadoopGis, MapReduce cannot access the constructed spatial index.

Recently, SpatialHadoop is developed in order to handle the limitations and drawbacks presented above in HadoopGis as well as Parallel SECONDO. However, since SpatialHadoop is based on Hadoop, on which it stores the data and

intermediate results on hard DISK, it has a high cost compared to GeoSpark, which is based on Spark in-memory. Hence, SpatialHadoop is much slower than GeoSpark.

Table 1. Distributed Spatial Data Analytics Frameworks.

Parameters	Spark	Hadoop
Data Storage	in memory	Disk
Intermediate results storage	in memory	Hard disk
Fault tolerance	guaranteed by RDD	data replication
Speed	faster due in-memory computation	relatively slower than spark
Os.support	Linux, Windows, Mac OS	Linux
Languages	Scala, Python, Java, R	Java
bottlenecks	Large memory consumption	frequent disk I/O

Table 2. Geospatial Data Analytics Platforms

Platforms	Hadoop-Based	Spark-Based	MapReduce-Based	support	Index	Spatial data	Speed
HadoopGis	X		MapReduce program cannot access the constructed spatial Index	Only applicable in uniform data distribution	Only uniform Grid		--
GeoSpark		X	X	limited community support Only Java, Scala	R-Tree, Quad-Tree	X	Much faster than Spatial-Hadoop
SpatialHadoop	X		X		R-Tree, R+-Tree, Grid file	X	--

4.2. The Effectiveness of Trajectory Data Processing

Even though many approaches have been proposed, some facets of trajectory query are not studied thoroughly or even paid attention to.

The concept of trajectory query is useful for increasing the usability of results and improving the experiences of users. Even though big trajectory data has been applied to various kinds of queries and mining tasks as surveyed in this paper, that is not enough to study the important kinds of query processing and data mining tasks related to user life. One important scenario is the similarity search of the frequent trajectories based on missing trajectory data. For such query, the data archived in TJDBs may not be complete. Therefore, an efficient index and algorithm should be necessary to handle such a problem. Unfortunately, none of the existing platforms, especially SpatialHadoop or GeoSpark, could be used to handle such a query issue. However, based on the open research directions presented in this study, we propose to use Spark – due to its cost and its speed – to develop effective query algorithms to efficiently traverse the large-scale TJDBs.

5. Conclusions

Trajectory data used in geographic information system GIS is helpful to users. For example, it can be used to find a frequent restaurant, hotel or route. This large-scale data is archived and organized in TJDBs. However, nowadays, the trajectory data processing exceeds the power of centralized approaches previously proposed.

Trajectory query is very necessary to extract the trajectory matching from TJDBs. Therefore, trajectory query processing is beneficial to many researchers in different domains. For example, individual citizens can understand his or her movement behavior through information analyzed from historical trajectories. Further, trajectory query processing greatly aids the public services through route recommendation and traffic information provided by business transportation, etc. However, as the massive movement object is archived through TJDBs, the text descriptive of these objects, and the paths on which these objects are connected are archived in TJDBs. For that reason, the query tasks in massive trajectory data are not insignificant.

Many researchers have proposed various methods and approaches to handle such challenges. This paper is a survey of an extensive collection of existing studies on trajectory data processing results as well as trajectory queries, including the mining trajectory query. It provides a definition of different distributed systems used and various existing approaches of trajectory query processing.

Acknowledgements

This paper was partially supported by NSFC grant U1509216,61472099, National Sci-Tech Support Plan 2015BAH10F01, the Scientific Research Foundation for the Returned Overseas Chinese Scholars of Heilongjiang Providence LC2016026 and MOE–Microsoft Key Laboratory of Natural Language Processing and Speech, Harbin Institute of Technology.

References

1. R. Agrawal and R. Srikant, "Mining Sequential Patterns," in *Proceedings of the 11th International Conference Data Engineering (ICDE)*, pp. 3-14, 1995.
2. E. Ahmed and M.F Mohamed, "A Demonstration of SpatialHadoop: An Efficient MapReduce Framework for Spatial Data," in *Proceedings of VLDB Endrow*, pp. 1230-1233, August 2013.
3. A. Aji and F. Wang, "High Performance Spatial Query Processing for Large Scale Scientific Data," in *Proceedings of the SIGMOD/PODS PhD Symposium*, pp. 9-14. New York, NY, USA, 2012.
4. A. Aji, F. Wang, H. Vo, R. Lee, R., Q. Liu., X. Zhang, and J. Saltz, "Hadoop GIS: A High Performance Spatial Data Warehousing System over Mapreduce," in *Proceedings of the VLDB Endow*, pp. 1009-1020, August 2013.
5. M. Anna, P. Fabio., T. Roberto, and G. Fosca "WhereNext: A Location Predictor on Trajectory Pattern Mining," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 637-646, New York, NY, USA, 2009.
6. T. Ashish, S. Joydeep, J. Namit, S. Shao, C. Prasad, A. Suresh, L. Hao, W. Pete, and M. Raghotham "Hive: A Warehousing Solution over a Map-reduce Framework," in *Proceedings of the. VLDB Endow*, pp. 1626-1629, August 2009.
7. L. Avinash and M. Prashant "Cassandra: A Decentralized Structured Storage System" in *SIGOPS Oper. Syst. Rev.* (ACM), vol. 44 pp. 35-40, 2010.
8. A. Azza, B-P. Kamil, A. Daniel, A. Silberschatz, and R. Alexander. "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads," in *Proceedings of VLDB* Endowment), pp. 922-933, 2009.
9. A. Azza, B-P. Kamil, A. Daniel, A. Silberschatz, and R. Alexander. "HadoopDB in Action: Building Real World Applications," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pp. 1111-1114, New York, NY, USA, 2010.
10. Y. Bin, M. Qiang, Q. Weining, and Z. Aoying, "Truster: Trajectory Data Processing on Clusters" (demo paper), *In DASFAA*, pp. 768-771. 2009.
11. O. Christopher, R. Benjamin., S. Utkarsh., K. Ravi, and T. Andrew, "Pig Latin: A Not-So-Foreign Language for Data Processing," in *Proceedings of International Conference on Management of Data the ACM SIGMOD*. pp. 1099-1110, New York, NY, USA, 2008.
12. H. Chunchun, K. Xionghua, L. Nianxue, and Z Qiansheng, "Parallel Clustering of Big Data of Spatio-Temporal Trajectory," in *Proceedings of the 11th International Conference on Natural Computation (ICNC), pp.* 769-774, 2015.
13. X. Dawen, W. Binfeng., L. Yantao, R. Zhuobo, and Z. Zhang, "An Efficient MapReduce-Based Parallel Clustering Algorithm for Distributed Traffic Subarea Division," *Discrete Dynamics in Nature and Society*, vol. 18, *2015*.
14. Z. Deng and Y. Bai, "Floating Car Data Processing Model based on Hadoop-GIS Tools," in *Proceedings of the Fifth International Conference on Agro-Geoinformatics*, pp. 1-4, July 2016.
15. C. Fay, D. Jeffrey., G. Sanjay, H.C. Wilson, W.A. Deborah, B. Mike, C. Tushar, F. Andrew, and G.E. Robert, "BigTable: A Distributed Storage System for Structured Data," *ACM Trans. Computer. System*., vol. 26, pp. 1-26, Jun 2008.
16. Q. Hongjiang, G. Rong, Y. Chunfeng., and H. Yihua, "YAFIM: A Parallel Frequent Itemset Mining Algorithm with Spark," *IEEE International Parallel Distributed Processing Symposium Workshops*, pp. 1664-1671, May 2014.
17. A. H. Htoo, G. Long, and T. Kian-Lee, "Mining Sub-trajectory Cliques to Find Frequent Routes," in *Proceedings of the 13th International Symposium on Advances in Spatial and Temporal Databases,* Vol.8098, New York, NY, USA, 92-109, 2013.
18. D. Jeffrey, G. Sanjay "MapReduce: Simplified Data Processing on Large Clusters," *Commun ACM*, vol. 51, pp. 107-113, January 2008.
19. Z. Jerry and P. Jelena, "MapReduce: The programming Model and Practice," SIGMETRICS Tutorial, 2009.
20. Y. Jia, W. Jinxuan, and S. Mohamed, "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,* New York, NY, USA, 2015.
21. L. Jiamin Liu and R.H. Güting, "Parallel Secondo: Boosting Database Engines with Hadoop," in *Proceedings of the IEEE 18th International Conference on Parallel and Distributed Systems*, pp. 738-743, December 2012.

22. L. Jiamin Liu and R.H. Güting," Parallel Secondo: A Practical System for Large-Scale Processing of Moving Objects," in *Proceedings of the IEEE 30th International Conference on Data Engineering*, pp. 1190-1193, Marsh, 2014.
23. G. Jian and R. Yong-gong, "Research on Improved A Priori Algorithm Based on Coding and MapReduce," in *Proceedings of the 10th Web Information System and Application Conference,* pp. 294-299, 2013.
24. H. Jian, L. Chunwei, and Q. Jinhui . "Developing Map Matching Algorithm for Transportation Data Center," in *Proceedings of the Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing,* pp. 167-170, 2014.
25. Z. Jianting, "Towards Personal High-Performance Geospatial Computing (HPC-G): Perspectives and a Case Study," in *Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems,* pp. 3-10, New York, NY, USA, 2010.
26. E. Masciari, S. Gao, and Z. Carlo, "Sequential Pattern Mining from Trajectory Data," in P*roceedings of the 17th International Database Engineering Applications Symposium*, pp. 162-167, New York, NY, USA, 2013.
27. Z. Matei and C. Mosharaf, "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation,* pp. 2-2, Berkeley, CA, USENIX Association, 2012.
28. I. Michael, B. Mihai, Y. Yuan, B. Andrew, and F. Dennis, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," *EuroSys.* 2007.
29. M. Mikolaj, "Prediction of Moving Object Location Based on Frequent Trajectories," in *Proceedings of the 21st International Conference on Computer and Information Sciences*, pp. 583-592, Berlin, 2006.
30. M. Mikolaj, "Mining Frequent Trajectories of Moving Objects for Location Prediction," in *Proceedings of the Fifth International Conference on Machine Learning and Data Mining in Pattern Recognition,* Vol. 4571 *of lecture notes in Computer science*, pp. 667-680. 2007.
31. Z. J. Mohammed, P. Srinivasan, O. Mitsunori, and L. Wei, "New Algorithms for Fast Discovery of Association Rules," in *Proceedings of the third International Conference on Knowledge Discovery and Data Mining,* pp283-286, Newport, Beach, 1997.
32. S. Natalijia and S. Dragan, "Processing and Analysis of Big Trajectory Data using MapReduce," *Facta Universitatis, Series: Automatic Control and Robotics,* vol. *14*, pp. 19-27, 2015.
33. L. Ning, Z.li, H. Qing, and S. ZhongZhi, "Parallel Implementation of Apriori Algorithm Based on MapReduce," in *Proceedings of the 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel Distributed Computing*, pp. 236-241, 2012.
34. H. Patrick, K. Mahadev, J.P Flavio, and R. Benjamin, "ZooKeeper: Wait-Free Coordination for Internet-Scale Systems," in *Proceedings of the USENIX Annual Technical Conference,* pp. 11-11, Berkeley, CA: USENIX Association, 2010.
35. M. Qiang., Y. Bin, Q. Weining, and Z. Aoying, "Query Processing of Massive Trajectory Data Based on Mapreduce," in *Proceedings of the First International Workshop on Cloud Data Management*. pp. 9-16, New York, NY, USA,2009.
36. G. H. Ralf, B. Thomas, and D. Christian "SECONDO: A Platform for Moving Objects Database Research and for Publishing and Integrating Research Implementations," IEEE Data Eng. *Bull.,* Vol. *33*, pp. 56-63, 2010.
37. G. Sanjay, G. Howard, and L. Shun-Tak, "The Google File System," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*. pp. 29-43, New York, NY, USA, 2003.
38. R. Sanjay., K. Manohar, and K. Arti, "R-Apriori: An Efficient Apriori Based Algorithm on Spark," in *Proceedings of the 8th Workshop on Ph.D. Workshop in Information and Knowledge Management*. pp. 27-34, New York, NY, USA, 2015.
39. Z. Shubin, H. Jizhong, L. Zhiyong, W. Kai, and X. Zhiyong, "SJMR: Parallelizing Spatial Join with MapReduce on Clusters," in *Proceedings of the IEEE International Conference on Cluster Computing and Workshops,* pp. 1-8, 2009.
40. T. Suryakanthi and V. S. J. Pallapolu. "A Comparative Study on Performance of Hadoop File System with MapR File System to Process Big Data Records," in *IJCSI International Journal of Computer Science Issues*, vol. *13*, pp. 129-140, 2016.
41. K. Wang, J. Han, B. Tu, J. Dai, W. Zhou, and X. Song, "Accelerating Spatial Data Processing with MapReduce," in *Proceedings of the IEEE 16th International Conference on Parallel and Distributed Systems*, pp. 229-236, December 2010.
42. X. Wang., X. Liu, B. Liu, E. N. de Souza, and S. Matwin, "Vessel Route Anomaly Detection with Hadoop MapReduce," in *Proceedings of the. IEEE International Conference on Big Data*, pp. 25-30, October 2014.
43. E. Wiam, W. Ali, and A.M. Adel, "An Investigation of Parallel Road Map Inference from Big GPS Traces Data," Procedia Computer Science, vol. 53, pp. 131-140, 2015.
44. Y. Xian, C. Xu, and Y. Liu. "Implementing Trajectory Data Stream Analysis in Parallel," in Proceedings *of the IEEE International Conference on Big Data*, pp. 2431-2436, 2016.
45. D. Xie, L. Feifei, Y. Bin, L. Gefei, Z. Liang, and G. Minyi. "Simba: Efficient In-Memory Spatial Analytics," in *Proceedings of the International Conference on Management of Data.* New York, NY, USA, 2016.
46. H. Yang, D. Ali, H. Ruey-Lung, and P. D. Stott, "Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters," in *Proceedings of the International Conference on Management of Data ACM SIGMOD.* New York, NY, USA, 2007.
47. L. Yanhua, C. Chi-Yin, D. Ke, Y. Mingxuan, Z. Jia, Z. Jia-Dong, Y. Qiang, and Z. Zhi-Li, "Sampling Big Trajectory Data," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 941-950, New York, NY, USA, ACM.

48. K. Yongchul, N. Dylan, G.P, Jeffrey, B. Magdalena, H. Bill Howe, and L. Sarah. "Scalable Clustering Algorithm for N-Body Simulations in a Shared-Nothing Cluster," in *Proceedings of the 22nd SSDBM Conference*, 2010.
49. Z. Yuanchun, Z. Yang, G. Yong, X. Zhenghua, F. Yanjie, G. Danhuai, S. Jing, Z. Tiangang, W. Xuezhi, and Li. Jianhui., "An Efficient Data Processing Framework for Mining the Massive Trajectory of Moving Objects," *Computers Environment and Urban Systems*, Vol. 61, pp. 129-140, 2017.

Amina Belhassena is a PhD. student at Harbin institute of technology. He received her Masters of Technology in Computer science from Abou bakr belkaid Tlemcen university, Algeria. His research interest includes massive data computing, data mining, large-scale data management, data indexing.

HongZhi Wang was born in 1978, male, PHD. He is a professor and doctoral supervisor at Harbin Institute of Technology. His research area is data management, including data quality, XML data management and graph management. He has published more than 100 papers in refereed journals and conferences. He is a recipient of the outstanding dissertation award of CCF, Microsoft Fellow and IBM PhD Fellowship.