JUST-Traj: A Distributed and Holistic Trajectory Data Management System (Demo Paper)

Huajun He[1,2], Ruiyuan Li[2*], Jie Bao[2], Tianrui Li[1], Yu Zheng[1,2]
[1]Southwest Jiaotong University, Chengdu, China [2]JD Intelligent Cities Research, Beijing, China
hehuajun@my.swjtu.edu.cn,trli@swjtu.edu.cn,{liruiyuan3,baojie3,zheng.yu}@jd.com

ABSTRACT

With the rapid development of the Internet of Things (IoT), massive trajectories have been generated. Trajectory data is beneficial for many urban applications. This demo presents a holistic trajectory data management system based on distributed platforms, such as Spark and HBase. It provides a variety of indexes to support various spatio-temporal queries and analyses on massive trajectories efficiently. Besides, it provides a convenient SQL engine to execute all operations (storage, queries, analyses) through a SQL-like statement. Finally, we open a web portal for developers, and demonstrate different operations in the portal.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**; *Query languages for non-relational engines*; Database query processing.

KEYWORDS

trajectory management, spatio-temporal query, trajectory analysis

ACM Reference Format:
Huajun He, Ruiyuan Li, Jie Bao, Tianrui Li, Yu Zheng. 2018. JUST-Traj: A Distributed and Holistic Trajectory Data Management System (Demo Paper). In *ACM SIGSPATIAL'21, November 02–05, 2021, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Various sensing devices and applications have collected massive trajectories of moving objects in recent years. For example, more than 1TB GPS logs are generated by over 60,000 couriers of JingDong each day [14], and T-Drive [16] contains 790 million trajectories generated in Beijing over only three months. Trajectory data is beneficial for many urban applications, e.g., traffic planning [7], reachability analysis [8], and epidemic prevention [6]. Taking advantage of distributed platforms is one of the best ways to manage large-scale trajectory data efficiently.

Existing works. In the last decade, existing works [1, 3, 4, 12, 15] leverage distributed computing platforms, e.g., Hadoop and Spark, to query and analyze massive trajectories. First, they adopt a strategy (e.g., STR) to assign trajectories into partitions. Then, they build a local index in each partition and a global index over all partitions to efficiently support spatio-temporal queries and analyses. However, they are arduous to support real-time data updates because they may spend much time readjusting the index structure and re-balancing the partitions when inserting a lot of new data. Thus, they are hard to scale up. The distributed NoSQL (Not Only SQL) data stores, such as HBase, are widely used to manage massive data in the disk. It can query data from the vast dataset efficiently and re-balance data nodes automatically. However, they do not support trajectory data analyses natively. Besides, applications always need various queries and analyses, which may require many operations on different platforms, e.g., querying data from HBase and analyzing data on Spark, struggling for a convenient way.

Our solution. Building on our previous works (i.e., JUST [9] and TrajMesa [10, 11]), we develop a distributed and holistic trajectory data management system, namely JUST-Traj. JUST provides a unified platform based on the Spark and NoSQL data store, which can provide spatio-temporal data queries and analytics through a convenient SQL engine. TrajMesa provides three spatio-temporal indexes to efficiently store and query trajectories on a NoSQL data store (i.e., HBase).

The advantages of our system are summarized as follows:

- JUST-Traj is a distributed and holistic system with an efficient management of massive trajectory data.
- JUST-Traj provides a complete SQL engine to conveniently operate (i.e., store, query, analyze) massive trajectories.
- We have implemented JUST-Traj and provide an online system to developers [5]. To the best of our knowledge, JUST-Traj is the first full-fledged (i.e., supporting storage, query, analytics, and SQL engine) online system for big trajectory data management.

The rest of this paper is organized as follows. Section 2 gives an overview of JUST-Traj. Section 3 stores trajectories into the NoSQL database. Section 4 introduces queries provided by JUST-Traj. Section 5 shows analytics of JUST-Traj. Section 6 describes the SQL engine. Section 7 gives the demonstration.

2 OVERVIEW

Figure 1 gives an overview of JUST-Traj, which contains four core components: (1) **storage** (Section 3), JUST-Traj stores trajectories into the NoSQL database by three steps, i.e., pre-processing, indexing, and storing; (2) **query** (Section 4), JUST-Traj provides many useful spatio-temporal queries on trajectories; (3) **analytics** (Section 5) provides many useful analysis operations for urban applications, e.g., processing, aggregation, stay point detection, clustering,

*Ruiyuan Li is the corresponding author

ACM SIGSPATIAL'21, November 02–05, 2021, Beijing, China

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Figure 1: Overview of JUST-Traj.

and close-contacts tracking; (4) SQL engine (Section 6) implements a complete SQL engine with many out-of-the-box operations preset, based on which all operations (i.e., storage, query and analytics) can be performed through a SQL-like query statement.

3 STORAGE

As shown in Figure 1 (storage), JUST-Traj pre-processes (Section 3.1) and indexes (Section 3.2) raw trajectories in Spark, then stores (Section 3.3) massive trajectories into HBase.

3.1 Pre-processing.

It is essential to pre-process trajectory, as the data noise and sampling rate in the raw GPS logs may a ect the accuracy and performance of applications. JUST-Traj supports four frequently-used operations to process trajectories, i.e., noise ltering, segmentation, interpolation, and map matching. Notably, the details of pre-processing can refer to our previous work [13].

Noise Filtering lters abnormal GPS logs, e.g., the points drift signi cantly out of a trajectory, as the inescapable error of many GPS terminals. If we do not remove the error points in the trajectory, applications may su er problems in data analysis tasks.

Segmentation breaks a trajectory into several segments. One terminal could generate a large number of GPS points without interruption, but only a part of points will be used in querying and analyzing. Thus, segmentation could reduce the computational complexity when executing data analysis tasks.

Interpolation inserts new points into a trajectory, as GPS terminals may neglect some important logs (e.g., the battery is low).

Map Matching projects a raw trajectory onto the road network. Therefore, it is essential for many applications based on the road network, e.g., tra c ow prediction and reachability query.

3.2 Indexing.

Indexing is vital for spatio-temporal queries, which can improve the e ciency of extracting data from the database. For example, JUST-Traj provides three spatio-temporal indexes for trajectories. (1) XZ2 , which is a ne-grained index for e ciently querying the trajectory from the database. It uses an index space of XZ-Ordering (XZ2) [2] to represent the minimum bounding rectangle (MBR) of a trajectory and a position code to describe the shape of it; (2) XZT . A trajectory has a time range from the start point to the end point. We index the time range of a trajectory by the temporal range index (XZT) proposed in TrajMesa [10], which helps JUST-Traj to query trajectories within a given time range; (3) XZ2+T, which is a spatio-temporal index for querying trajectories by a given spatio-temporal range. We rst split the time dimension into multiple disjoint time periods, then construct an individual XZ2 index in each time period.

More details of indexes can refer to our previous work [10].

3.3 Storing.

We store the cleaned and indexed trajectories into the NoSQL database using the form of key-value. First, we generate di erent kinds of keys that contain the spatio-temporal information of a trajectory, as shown in Figure 1 (storage) (Storing). Then, we compress the value of a trajectory into one column, which reduces the storage size and the I/O overhead. After that, we store the key-value pairs of the trajectory into the indexing tables for the later queries. We use the eld traj to represent the value of a trajectory and execute query and analysis operations on that eld.

4 QUERY

In this section, we introduce the fundamental trajectory queries provided by JUST-Traj.

ID Temporal Query. It retrieves trajectories by an object ID and a temporal range, which could help managers to know the detailed trajectory of a particular driver in a given temporal range, e.g., nding the trajectory generated by the taxi 1001 from 8:00 to 10:00 in a day.

Spatial Range Query. It nds trajectories by their relationship with a given spatial range, e.g., nding all trajectories that traversed or were fully contained in Times Square.

Spatio-temporal Query. It searches trajectories by a spatio-temporal range, e.g., nding all trajectories passing a railway station area

Figure 2: The Architecture of SQL Engine.

from 15:00 to 17:00 in a day.
Other Queries. JUST-Traj supports a variety of special queries for trajectories, e.g.,similarity query nds trajectories similar to a given trajectory andkNN query nds top-k similar trajectories.

5 ANALYTICS

JUST-Traj provides many out-of-the-box data analysis functions for trajectories, which facilitates the development of applications. Figure 1(Analytics) shows ve popular trajectory data analyses provided by JUST-Traj, i.e.,
Processing. Although we can pre-process trajectories before storing, parameters of algorithms could be adjusted when analyzing. That is, JUST-Traj also supports the processing in analytics stage;
Aggregation. JUST-Traj provides many aggregation operations, e.g.,<0G¹º, <8=¹º, 2>D=tC;
Stay Point Detection. Moving objects tend to stay due to certain events, such as vehicles staying for refueling, couriers staying for delivery. By analyzing the place that a moving object stays, we can infer some places of interest, e.g., the delivery addresses;
Clustering. It is one of the basic methods to explore the movement patterns of groups;
Close-contacts tracking. It nds people who had close contact with an abnormal person. It is vital for many applications, e.g., epidemic prevention [6] and companion detection.

6 SQL ENGINE

It is troublesome for users to execute operations on di erent platforms, e.g., querying data from HBase but analyzing data in Spark. JUST-Traj implements a complete SQL engine with many out-of-the-box operations preset by extending Apache Calcite (Section 6.1). Based on that, all operations (i.e., storage, query, and analytics) can be performed through a SQL-like query statement (Section 6.2).

6.1 Architecture

Figure 2 displays the architecture of our SQL engine. JUST-Traj provides aDriver for developers to interact with the SQL engine. The Serversends and receives data through the JDBC or RESTful API to the Driver. We parseand validatethe SQL by integrating the SQL syntax of Section 6.2 into Antlr4. After that, we generate the regular or spatio-temporal operator expressions. Then, JUST-Traj improves the CBO and RBO of Calcite using spatio-temporal indexes to optimize the queries. Finally, JUST-Traj can generate a scanner to extract trajectories from HBase or through Spark SQL to execute queries and analyses on massive trajectories. Notably, JUST-Traj puts all spatio-temporal operations into the coprocessor of HBase, which signi cantly improves the query e ciency.

6.2 SQL

Our SQL engine consists of four types of statements to operate the database.
(1) DDL, which is the data de nition language to create and drop tables, e.g., JUST-Traj uses the following statement to create a trajectory table:

```
CREATE TABLE <table name> (<field name> Trajectory)
WITH (<key=values>);
```

where < eld name> is the eld name of a trajectory and <key-values> sets the con guration (e.g., enable or disable spatio-temporal indexes, JUST-Traj acquiescently enables all indexes) of the table.
(2) DML. It loads data from multiple sources into JUST-Traj. For example, we can load data using the following statement:

```
LOAD <source type>:<file path> TO JUST:<table name>
CONFIG {<the field mapping relationship>};
```

where <source type> could be HDFS, HIVE, KAFKA.CONFIGprovides the eld mapping from the source to the JUST-Traj table.
(3) DQL. It selects trajectories from tables. JUST-Traj provides spatial or spatio-temporal queriesspatial range query, spatio-temporal range query, ID temporal query, similarity query, kNN query.For example, the SQL of thspatial range queryis as follows:

```
SELECT * FROM <table name>
WHERE st_within(traj, st_makeBBox(lng1,lat1, lng2, lat2));
```

wherest_makeBBoxis a spatial range formed by two points (lng1, lat1) and (lng2, lat2)
(4) DAL is a particular statement provided by JUST-Traj for trajectory data analyses, e.g., processing, aggregation, stay point detection, clustering, close-contacts detection. The SQL statement of DAL is as follows:

```
SELECT <analyzing operation>(traj, {<parameters>})
FROM <table name>;
```

where,analyzing operationis the name of analysis,parameterset the corresponding parameters. Section 7.2 gives two examples.

7 DEMONSTRATION

We open an online web portal for executing JUST-Traj SQL (Section 7.1). We demonstrate JUST-Traj using trajectories from lorries of Guangzhou, China, and taxis of Guiyang, China. Two holistic scenarios are demonstrated in Section 7.2.

7.1 Web Portal

As shown in Figure 3, the web portal of JUST-Traj has three panels: (1) table panel, which manages the created tables; (2) SQL panel, which provides an SQL editor; and (3) result panel, which visualizes

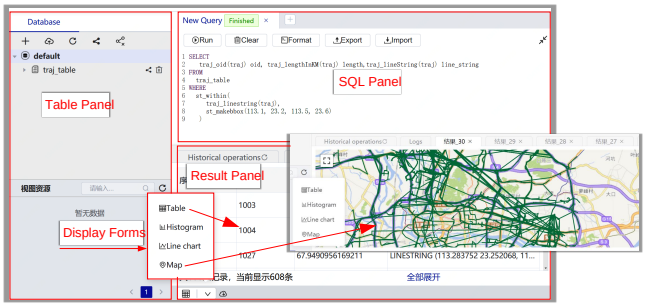


Figure 3: The Web Portal of JUST-Traj [5] (http://just-traj.urban-computing.com).

(a) Raw Trajectories (b) Stay Points

Figure 4: The Result of Stay Point Detection.

the result by multiple display forms, i.e., table view, chart view (i.e., histogram and line chart), and map view.

7.2 Scenarios

7.2.1 Storage. (1) We first create a trajectory table, namely *traj_table*:

```
CREATE TABLE traj_table (traj Trajectory);
```

where *traj* is the field name that denotes a trajectory in JUST-Traj. (2) Then, we load trajectories from HDFS into JUST-Traj:

```
LOAD HDFS:'/trajectories' to JUST:traj_table (
  oid 0,
  time to_timestamp(3),
  point st_makePoint(1, 2)
);
```

where '/trajectories' is the path of trajectories, lines from 2 to 4 are the field mappings.

7.2.2 Stay Point Detection. In this scenario, we detect stay points from the results of a spatio-temporal query. A stay point is the location where a driver stays over a given time threshold (*minStayTimeInSecond*), and the location is a spatial region whose maximum distance is not greater than a distance threshold (*maxStayDisInMeter*). The underlying locations of stay points could be the delivery addresses. The details of more parameters have been introduced in our handbook [5]. The SQL is as follows:

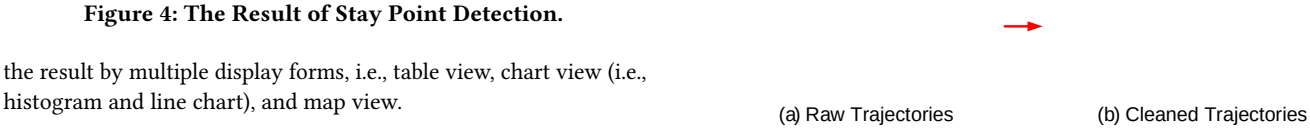
```
SELECT st_trajStayPoint(traj,
  '{ "maxStayDistInMeter": 10,
     "minStayTimeInSecond": 60}')
FROM
  traj_table
WHERE
  st_within(traj_linestring(traj),
      st_makeBBox(113.0, 23.0, 113.5, 23.6))
  and traj_startTime(traj) >= '2014-03-13 07:04:51'
  and traj_endTime(traj) <= '2014-03-16 08:04:51';
```

Lines from 7 to 10 take a spatio-temporal range to query trajectories from the database. Lines from 1 to 3 execute the *Stay Point Detection* operation on the extracted trajectories, where lines from 2 to 3 are parameters of *Stay Point Detection*. Figure 4 displays the results.

7.2.3 Noise Filtering. In this scenario, we define the point whose speed exceeds the maximum limited speed (*maxSpeedMeterPerSecond*) as a noise point. More parameters of noise filtering can refer to our handbook [5]. The SQL is as follows:

```
SELECT st_trajNoiseFilter(traj,
  '{ "maxSpeedMeterPerSecond": 20.0}')
FROM
  traj_table
WHERE
  traj_oid(traj) = '1197404443')
  and traj_startTime(traj) >= '2018-07-03 14:33:27'
  and traj_endTime(traj) <= '2018-08-03 14:33:27';
```

Lines from 6 to 8 take a ID temporal query to extract trajectories from the database. Lines from 1 to 3 execute the *Noise Filtering* operation on the extracted trajectories. Figure 5 shows the results.



(a) Raw Trajectories (b) Cleaned Trajectories

Figure 5: The Result of Noise Filtering.

REFERENCES

[1] Louai Alarabi. 2018. Summit: a scalable system for massive trajectory data management. In *SIGSPATIAL*. 612–613.

[2] Christian BÖxhm, Gerald Klump, and Hans-Peter Kriegel. 1999. Xz-ordering: A space-filling curve for objects with spatial extension. In *SSTD*. Springer, 75–90.

[3] Xin Ding, Lu Chen, Yunjun Gao, Christian S Jensen, and Hujun Bao. 2018. Ultraman: a unified platform for big trajectory data management and analytics. *Proceedings of the VLDB Endowment* 11, 7 (2018), 787–799.

[4] Ziquan Fang, Lu Chen, Yunjun Gao, Lu Pan, and Christian S Jensen. 2021. Dragoon: a hybrid and efficient big trajectory management system for offline and online analytics. *The VLDB Journal* 30, 2 (2021), 287–310.

[5] Huajun He. 2021. JUST-Traj. http://just-traj.urban-computing.com/.

[6] Huajun He, Ruiyuan Li, Rubin Wang, Jie Bao, Yu Zheng, and Tianrui Li. 2020. Efficient suspected infected crowds detection based on spatio-temporal trajectories. *arXiv preprint arXiv:2004.06653* (2020).

[7] Tianfu He, Jie Bao, Ruiyuan Li, Sijie Ruan, Yanhua Li, Chao Tian, and Yu Zheng. 2018. Detecting Vehicle Illegal Parking Events using Sharing Bikes' Trajectories.. In *SIGKDD*. 340–349.

[8] Ruiyuan Li, Jie Bao, Huajun He, Sijie Ruan, Tianfu He, Liang Hong, Zhongyuan Jiang, and Yu Zheng. 2020. Discovering Real-Time Reachable Area Using Trajectory Connections. In *DASFAA*. Springer, 36–53.

[9] Ruiyuan Li, Huajun He, Rubin Wang, Yuchuan Huang, Junwen Liu, Sijie Ruan, Tianfu He, Jie Bao, and Yu Zheng. 2020. Just: Jd urban spatio-temporal data engine. In *ICDE*. IEEE, 1558–1569.

[10] Ruiyuan Li, Huajun He, Rubin Wang, Sijie Ruan, Tianfu He, Jie Bao, Junbo Zhang, Liang Hong, and Yu Zheng. 2021. TrajMesa: A Distributed NoSQL-Based Trajectory Data Management System. *TKDE* (2021), 1–1. https://doi.org/10.1109/TKDE.2021.3079880

[11] Ruiyuan Li, Huajun He, Rubin Wang, Sijie Ruan, Yuan Sui, Jie Bao, and Yu Zheng. 2020. Trajmesa: A distributed nosql storage engine for big trajectory data. In *ICDE*. IEEE, 2002–2005.

[12] Ruiyuan Li, Sijie Ruan, Jie Bao, and Yu Zheng. 2017. A cloud-based trajectory data management system. In *SIGSPATIAL*. 1–4.

[13] Sijie Ruan, Ruiyuan Li, Jie Bao, Tianfu He, and Yu Zheng. 2018. Cloudtp: A cloud-based flexible trajectory preprocessing framework. In *ICDE*. IEEE, 1601–1604.

[14] Sijie Ruan, Zi Xiong, Cheng Long, Yiheng Chen, Jie Bao, Tianfu He, Ruiyuan Li, Shengnan Wu, Zhongyuan Jiang, and Yu Zheng. 2020. Doing in One Go: Delivery Time Inference Based on Couriers' Trajectories. In *SIGKDD*. 2813–2821.

[15] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. Dita: Distributed in-memory trajectory analytics. In *ICDE*. 725–740.

[16] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *SIGKDD*. 316–324.