TREST: A Hadoop Based Distributed Mobile Trajectory Retrieval System

Jianming Lv
School of Computer Science and Engineering
South China University of Technology
Guangzhou, 510006, China
Email: jmlv@scut.edu.cn

Xintong Wang
School of Computer Science and Engineering
South China University of Technology
Guangzhou, 510006, China
Email: w.xintong@mail.scut.edu.cn

Fengtao Huang
School of Computer Science and Engineering
South China University of Technology
Guangzhou, 510006, China
Email:huang.ft@mail.scut.edu.cn

Junjie Yang
School of Computer Science and Engineering
South China University of Technology
Guangzhou, 510006, China
Email: yang.junjie@mail.scut.edu.cn

Tianfeng Wu
School of Computer Science and Engineering
South China University of Technology
Guangzhou, 510006, China
Email: wu.tianfeng@mail.scut.edu.cn

Qifa Yan
School of Computer Science and Engineering
South China University of Technology
Guangzhou, 510006, China
Email: yan.qifa@mail.scut.edu.cn

***Abstract*—Nowadays, mobile phones have prevailed in a great variety of applications, through which massive personal trajectories can be collected and analyzed to support many interesting location-based services. However, it is still challenging to efficiently store and retrieve this kind of spatio-temporal data, which has typical big-data feature with large size, streaming style and multiple data source. To address this issue, we develop a mobile trajectory retrieval system named TREST, which is based on the distributed Hadoop and HBase systems. TREST makes use of the horizontal expansion mechanism of Hadoop to store overwhelming spatio-temporal trajectories, and supports frequent incremental insertion of data stream. Meanwhile, TREST maps the spatio-temporal features of trajectories into the simple key-value schema of HBase to support fast retrieval. We also develop a prototype of TREST to manipulate the real mobile trajectory data set, which contains totally 104 million records collected by mobile service providers. Experiments on this data set show that TREST can efficiently support both Single-track and All-track retrieval within milliseconds on average.**

I. Introduction

With the popular use of mobile phones and mobile internet, huge amounts of personal trajectories of mobile users can be collected and analyzed every day by mobile service providers to support some interesting location based applications, such as optimization of mobile networks, intelligent tour-guiding system, smart driving system, customized tourist attractions recommendation, and etc. It is meaningful and challenging to build a scalable and efficient data management system to store and process this kind of spatio-temporal data with large and fast increasing size.

Recently, various tree indexing structures are proposed to improve the efficiency of trajectory retrieval, such as TPR*Tree[1], B+Tree [2] and grid-based retrieval[3]. Based on these indexing algorithms, some specific data management systems have been developed to store and process spatio-temporal trajectories, such as TrajStore[4] and PIST[5]. TrajStore partitions trajectories into spatio-temporal regions, and adopts an adaptive multi-level grid to support efficient look up. PIST, on the other hand, divides the entire storage space by grid optimization and establishes B+ tree temporal index in each sub-space to boost the performance outcomes. Both TrajStore and PIST take a non-distributed architecture and appear incapacity in manipulating very large trajectory data sets. Different from TrajStore and PIST, CloST [6] is based on the distributed system Hadoop to support better scalability. CloST divides the records according to the spatio-temporal properties, and organizes the partitions in a 3-level file directory tree on the distributed file system HDFS. Queries of CloST are executed in parallel by using the MapReduce of Hadoop. OceanST [7] is a recently proposed system based on Hadoop and adopts the similar data partition strategy of CloST. OceanST improves the search efficiency of CloST by introducing 3-dimensional spatio-temporal index in the block file. OceanST also replace the disk-based MapReduce procedures of CloST by fast memory-based Spark iterations. However, the file directory based partition techniques of both CloST and OCeanST makes these system inefficient to perform the all-object quereis[6], which retrieve all trajectories intersecting a given spatio-temporal range. The queries can cause scan operations on a lot of block files distributed in different file directories of HDFS. The time cost of these queries can be hundred of seconds [6].

To achieve both of the scalability and efficiency goals in processing mobile trajectories, we propose the Trajectory

Retrieval System (TREST) in this paper. Our main contributions are summarized as follows:

- We propose a Hadoop/HBase based lightweight index mechanism to support distributed and efficient trajectory retrieval. Similar with CloST and OceanST, TREST utilizes Hadoop to support horizontal scalability, but it directly maps trajectories into the simple key-value schema of the HBase, which is an efficient NoSQL database on Hadoop. The index structure of TREST is more lightweight than CloST and OceanST, and can achieve much better retrieval efficiency. The search time of TREST can be within milliseconds towards the real trajectory data set with 104 million records.
- We store the trajectory data in a redundant manner to enhance efficiency of different search tasks. The trajectory records are partitioned into the redundant HBase tables, each of which selects a specific row key to optimize the search efficiency.
- Based on TREST, we develop a distributed version of the criminal tracing system [2], which is our preceding research and is designed to identify criminals' mobile phones by retrieving mobile trajectories matched with a given moving pattern. Exhaustive experiments are also conducted on real trajectory data set collected from mobile base stations.

The rest of this paper is organized as follows. Section II reviews the background of Hadoop and HBase. Section III offers clear definitions of the trajectory retrieval problem. Section IV presents our proposed methods. Section V evaluates the performance of this system by implementing the prototype and conducting experiments on real data sets. We conclude the work in Section VI.

II. Background of Hadoop and HBase

Hadoop is an open-source distributed storage and parallel computing framework from the Apache Project. The data processed in Hadoop is stored and organized by the integrated Hadoop Distributed File System (HDFS), which provides large-scale linearly expandable distributed storage capability. HBase is a distributed NoSQL database developed on HDFS. Aided by the high throughput of HDFS, HBase can support reading and writing large scales of bundle data in real time. Moreover, HBase organizes the data into simple $< key, value >$ pairs and supports efficient key based search in a distributed manner. The data in a HBase table is logically partitioned into regions according to the keys of records and indexed in a distributed linear index tree. How to select proper keys is critical to optimize the retrieval performance of HBase.

III. Problem definition

A. Definition of Mobile Trajectories

In this paper, we focus on processing the mobile trajectory data collected from base stations of cellular mobile networks. As we know, mobile networks divide land area into the cells, each of which is supplied with radio service from one base station. When someone carries a mobile phone and moves in service areas, his trajectory can be recorded as a sequence of tuples $(U_i, TF_i, TE_i, C_i, A_i)$ like Table I. Each row of the table indicates a mobile phone U_i is located in the cell C_i from the time TF_i to TE_i . A_i means some attached information of the record, such as the type of the mobile phone. The formal definition of the mobile trajectory is given as follows:

Definition 1 (Spatio-temporal Segment) . If a mobile phone U_i is located at the cell C_i from the time TF_i to TE_i, we define $\delta_i =< U_i, TF_i, TE_i, C_i, A_i >$ as a spatio-temporal segment. Here A_i is the extension information of the record.

Definition 2 (Phone Trajectory). The trajectory of a mobile phone is denoted as a sequence of Spatio-temporal Segments: $\xi_i = \{\delta_{i1}, \delta_{i2}, ..., \delta_{in}\}$ ordered by the time.

In the real deployment of a mobile retrieval system, it is useful to search for the trajectories matching some specific Spatio-temporal pattern [2]. Similar with the research [2], we define the query pattern as a sequence of Spatio-temporal Points as follows.

Definition 3 (Spatio-temporal Point). The Spatio-temporal Point is defined as $\omega_i =< C_i, T_i >$, which means an event happens in the cell C_i at the time T_i.

Definition 4 (Spatio-temporal Pattern). The Spatio-temporal Pattern is defined as a sequence of Spatio-temporal Points:

$$\zeta_i = \{\omega_{i1}, \omega_{i2}, ..., \omega_{in}\}.$$

Definition 5 (Trajectory Matching) . A Phone Trajectory ξ_i is said to be matched with a spatio-temporal pattern ζ_j (denoted as $\xi_i \rhd \zeta_j$), if and only if the following condition is satisfied. For each Spatio-temporal Point $\omega_j =< C, T >\in \zeta_j$, there exists a Spatio-temporal Segment $\delta_{ik} =< U_{ik}, TF_{ik}, TE_{ik}, C, A_{ik} >\in \xi_i$ that has the following property: $TF_{ik} \leq T \leq TE_{ik}$.

Table I
Records of Phone Trajectories

U_k	TF_k	TF_k	C_k
U_1	2:20'05"	2:40'10"	C_1
U_1	2:40'11"	3:00'20"	C_2
U_2	2:25'08"	2:30'20"	C_2
U_2	2:31'06"	2:39'17"	C_9
U_3	2:26'09"	3:00'19"	C_5
U_4	2:27'10"	2:30'18"	C_2

For clarity, some important notations used in this paper arep listed in the Table II.

B. Trajectory Retrieval Tasks

The location-based retrieval of mobile trajectories can be classified into the following two categories: Single-track Retrieval and All-tracks Retrieval as follows.

Table II
SUMMARY OF NOTATIONS.

Notation	Description
$\delta_i =< U_i, TF_i, TE_i, C_i, A_i >$	A Spatio-temporal Segment that means the mobile phone U_i is located in the cell C_i from the time TF_i to TE_i. A_i is some extension information of the record.
$\xi_i = \{\delta_{i1}, \delta_{i2}, ..., \delta_{in}\}$	A Phone Trajectory.
$\omega_i =< T_i, C_i >$	A Spatio-temporal Point that means an event happens in the cell C_i at T_i.
$\zeta_i = \{\omega_{i1}, \omega_{i2}, ..., \omega_{in}\}$	A Spatio-temporal Pattern.

- **Single-track Retrieval:** Given a mobile phone ID U_x and a time range $[TF_x, TE_x]$, the task is to retrieve the trajectory of U_x starting from TF_x to TE_x. The returned trajectory is formulated as $\xi_x = \{\delta_{x1}, \delta_{x2}, ..., \delta_{xn}\}$. Specifically, for each Spatio-temporal Segment in the trajectory
$$\delta_{xi} =< U_x, TF_{xi}, TE_{xi}, C_{xi}, A_{xi} >,$$
it should be satisfied that $TF_x \leq TF_{xi} \leq TE_x$.
- **All-tracks Retrieval:** Given a Spatio-temporal Pattern $\zeta_i = \{\omega_{i1}, \omega_{i2}, ..., \omega_{in}\}$, the task is to search for all of the Phone Trajectories, which are matched with ζ_i.

IV. SYSTEM DESIGN

A. Storage Model

The TREST system consists of three layers as illustrated in Fig. 1.

The topmost Logical Layer serves as the logical view of the trajectory data. In this level, the data collection stored in TREST is organized into three logical HBase tables: Base, Traj, and Map. The schema of these tables are illustrated in the Table III. Each record in a HBase Table is uniquely identified by a row key. All records are stored as block files of HDFS, and are sorted according to their row keys. Specifically, the Base table contains the description information of base stations, and uses the cell id C_i as the row key to facilitate efficient search of base stations. lon_i is the longitude of a base station and lat_i is the latitude. S_i indicates some related description information.

The Traj table is a core structure to hold all information of Phone Trajectories. Each row of the table is corresponding to a Spatio-temporal Segment:
$$\delta_i =< U_i, TF_i, TE_i, C_i, A_i >.$$
The row key in the table is denoted as $\overline{U_i \bigoplus TF_i}$, which is the concatenation of U_i and TF_i. It is designed to support efficient Single-track Retrieval, which searches for someone's Spatio-temporal Segments within given time range. The Map table is a redundant table to hold necessary mapping information to enable fast All-track Retrieval tasks. The row key of the table is denoted as $\overline{C_i \bigoplus TF_i}$, which is the concatenation of C_i and TF_i. While performing All-track Retrieval, the records of the mobile phones at a specific spatio-temporal range can be retrieved efficiently. When a new spatio-temporal segment is recorded, it is inserted in the Traj table and the corresponding mapping information is inserted into the Map table.

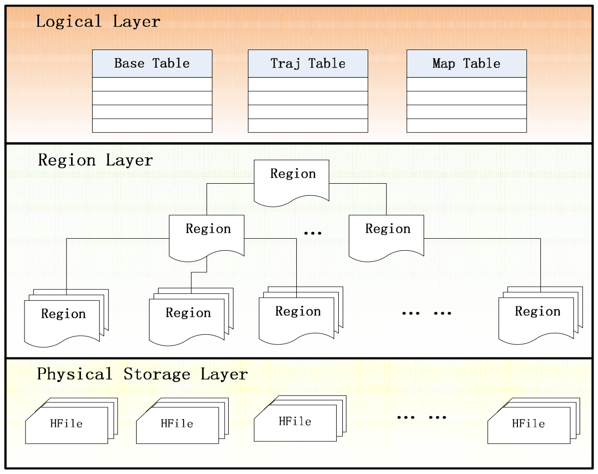


Figure 1. The storage architecture of TREST

Table III
HBASE TABLES

Table	Row Key	Columns
Base	C_i	lon_i, lat_i, S_i
Traj	$\overline{U_i \bigoplus TF_i}$	TE_i, C_i, A_i
Map	$\overline{C_i \bigoplus TF_i}$	TE_i, U_i

The middle layer is the Region layer, which partitions the HBase tables into regions according to their row keys. In HBase, the regions are basic units of balanced loading, and they are managed by region servers of HBase in a distributed manner. The regions are further organized as a multiple-level linear index tree based on the row keys of the records. The index can support efficient row key based search of the data. The bottommost Physical Storage Layer is the same as the persistence storage layer of HBase to store the data as HFiles in HDFS.

B. Single-track Retrieval

The algorithm to perform the Single-track Retrieval is illustrated in Algorithm 1. The search is executed on the Traj table. Given the mobile phone ID U_x and the time range $[TF_x, TF_y]$, the algorithm simply search the Traj table for the records, whose row keys are between $[\overline{U_x \bigoplus TF_x}$ and $\overline{U_x \bigoplus TE_x}]$. Each returned record is denoted as a matched Spatio-temporal Segment δ_j. After collecting all of these retrieved segments, we can obtain the target trajectory $\xi_x = \{\delta_j\}$.

Algorithm 1 Single-track Retrieval

```
Input: $U_x, TF_x, TE_x$
Output: The trajectory $\xi_x$ of the mobile phone $U_x$ during the time range $[TF_x, TE_x]$.
Method:
  1. $\xi_x \leftarrow \varnothing$
  2. $K_s \leftarrow \overline{U_x \bigoplus TF_x}$
  3. $K_e \leftarrow \overline{U_x \bigoplus TE_x}$
  4. $\Omega \leftarrow$ search the Traj table for the records whose row keys belong to $[K_s, K_e]$
  5. For each $(\overline{U_j \bigoplus TF_j}, TE_j, C_j, A_j) \in \Omega$ do
  6.     $\delta_j \leftarrow < U_j, TF_j, TE_j, C_j, A_j >$
  7.     $\xi_x \leftarrow \xi_x \cup \{\delta_j\}$
  8. Return $\xi_x$
```

C. All-track Retrieval

The All-track retrieval task is to search for all of the phone trajectories, which are matched with a given Spatio-temporal Pattern $\zeta = \{\omega_1, \omega_2, ..., \omega_n\}$. Here $\omega_i =< C_i, T_i >$. As defined in Section III.A, if a trajectory ξ_j is matched with ζ, it should have the following property. For each Spatio-temporal Point $\omega_i =< C_i, T_i >\in \zeta$, there must exist a Spatio-temporal Segment

$$\delta_k =< U_k, TF_k, TE_k, C_k, A_k >\in \xi_j \tag{1}$$

that satisfies:

$$TF_k \leq T_i \leq TE_k \tag{2}$$

and

$$C_k = C_i. \tag{3}$$

That means the trajectory ξ_j intersects the spatio-temporal range of each $\omega_i \in \zeta$. ω_i acts like a spatio-temporal constraint condition while performing the All-track Retrieval. Thus, As illustrated in Algorithm 2, we use each $\omega_i =< C_i, T_i >\in \zeta$ as a filter condition to search the Map table.

The parameter $\triangle$ of line 2 is the maximum timing interval between two adjacent Spatio-temporal Segments of a mobile phone. When a mobile phone is in work mode, it registers itself to the mobile network and reports its status to the nearby base station periodically. Each report message is then recorded as one Spatio-temporal segment recorded in mobile networks. The timing interval between two adjacent Spatio-temporal segments records varies from seconds to about half an hour. To ensure that all matched results can be retrieved, we set the maximum interval $\triangle$ as one hour. Because $\triangle$ is the maximum timing interval, according to Eq(2) we have:

$$T_i - \triangle \leq TF_k \leq T_i \tag{4}$$

Thus, we only need to search for the Spatio-temporal Segments satisfying both Eq(3) and Eq(4). As showed in line 6 of Algorithm 2, this can be done efficiently by searching the Map table for the row key between $\overline{C_i \bigoplus T_i'}$ and $\overline{C_i \bigoplus T_i}$, where $T_i' = T_i - \triangle$. The search result can be further filtered by checking whether $TE_k \geqslant T_i$ in line 8. In line 10, we collect the IDs of the mobile phones, whose trajectories intersect the spatio-temporal range of each $\omega_i \in \zeta$. In line 12, we perform the Single-track Retrieval to achieve the complete information of each matched trajectory.

Algorithm 2 All-tracks Retrieval

```
Input: A Spatio-temporal Pattern $\zeta = \{\omega_1, \omega_2, ..., \omega_n\}$, where $\omega_i =< C_i, T_i >$.
Output: A set of trajectories $\{\xi_j | \xi_j \triangleright \zeta\}$
  1. For each $\omega_i =< C_i, T_i >\in \zeta$ do
  2.     $T_i' \leftarrow T_i - \triangle$
  3.     $K_s \leftarrow \overline{C_i \bigoplus T_i'}$
  4.     $K_e \leftarrow \overline{C_i \bigoplus T_i}$
  5.     $\alpha_i \leftarrow \varnothing$
  6.     $\Omega \leftarrow$ search the Map table for the records whose row key belong to $[K_s, K_e]$
  7.     For each $(\overline{C_k \bigoplus TF_k}, TE_k, U_k) \in \Omega$ do
  8.         if $TE_k \geqslant T_i$ then
  9.             $\alpha_i \leftarrow \alpha_i \cup U_k$
  10.  $\alpha^* = \alpha_1 \cap \alpha_2 \cap ... \cap \alpha_n$
  11.  For each $U_k \in \alpha^*$ do
  12.      $\xi_k \leftarrow$ Single-track$(U_k, T_1, T_n)$
  13.  Return $\{\xi_k\}$
```

V. Evaluation

In this section, we conduct the experiments on the real trajectory data collected from base stations in the urban area of Guangzhou, China, from 0:00 am to 8:00am on Feb 2, 2013. The data set contains 104 million Spatio-temporal Segments of 5.9 million mobile phones located in 24,370 cells. We deploy TREST as a cluster of three interlinked PCs, each of which has single quad-core intel i5-3470 CPU of 3.20GHz, and 12GB memory. The operating system is Ubuntu 14.04 LTS (64-bit).

Based on above configuration, we evaluate TREST for the speed of incremental data insertion, the efficiency of Single-track and All-track Retrieval. Moreover, the visualized demonstration of the prototype is also proposed in this section.

A. Efficiency of data Insertion

Because the mobile trajectories are collected continuously every second, it is important for TREST to support efficient insertion of data stream. To evaluate the efficiency of data insertion on TREST, we divide the raw data containing 104 millions of Spatio-temporal Segments into the eight subsets with equal size, which are named $S_1 \sim S_8$. The records of $S_1 \sim S_8$ are inserted into TREST sequentially. The time consumed in the insertion of each subset is compared in Fig. 2, which shows that the time to insert the records of

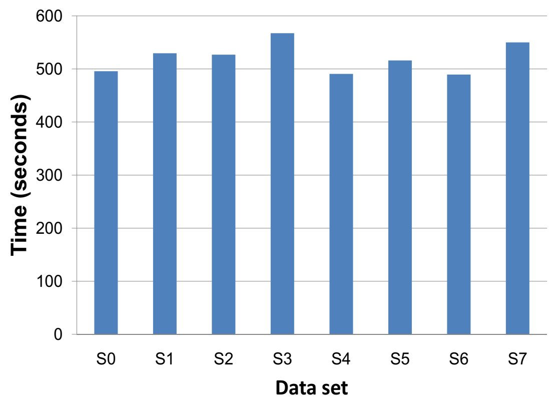


Figure 2. The time cost to insert records of different data sets.

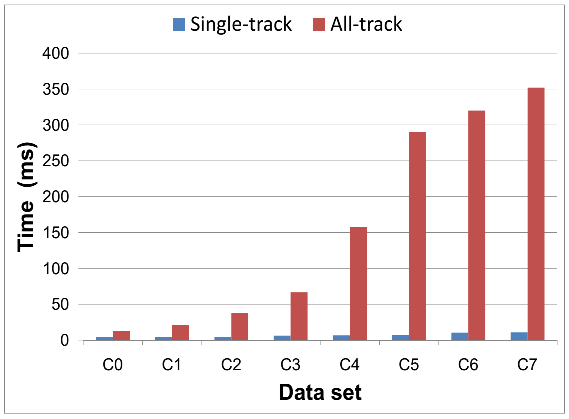


Figure 3. Retrieval time on the data sets with different size.

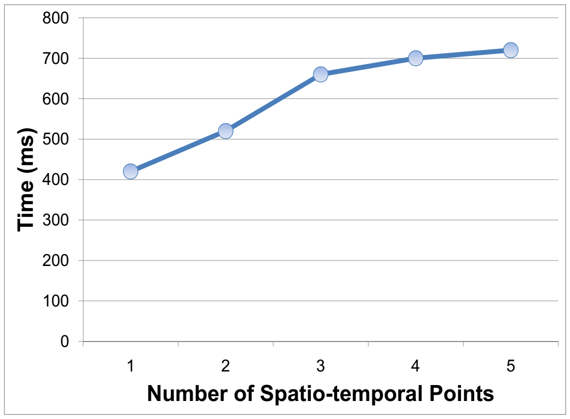


Figure 4. The time cost of All-track Retrieval vs the number of Spatio-temporal Points in query pattern.

the 8 subsets are almost equal. The size of the data stored in TREST affects the efficiency of inserting new records slightly. The average insertion speed is about 1.9M/s on the clusters of three normal PCs. Due to the horizontal scalability of HBase, increasing the number of machines or adopting more powerful servers may increase the throughput greatly in real deployment.

B. *Efficiency of Retrieval*

In order to evaluate the retrieval efficiency in different scales of data sets, we divide the raw data into eight subsets $T_0 \sim T_7$, each of which contains one hour of records as illustrated in Table IV. Then we compose the data sets of $C_0 \sim C_7$ by accumulating the subsets $T_0 \sim T_7$. Specifically, C_i is the concatenation of the subsets, T_0,T_1,...,T_i. Both Single-track Retrieval and All-track Retrieval are evaluated on these data sets, the detailed properties of which are listed in Table IV. For each Single-track Retrieval task, a mobile phone ID is selected from the data set randomly and an one-hour time range $[TF, TE]$ is randomly selected from the 0:00 am to 8:00am at Feb 2,2013. The task is to search for the trajectories of the specific mobile phone from the time TF to TE. On the other hand, the All-track Retrieval task is constructed in a similar randomized manner. Each Spatio-temporal Pattern of a retrieval task is composed of the Spatio-temporal Points, each of which is sampled from a randomly selected trajectory in the data set. The number of the Spatio-temporal Points in a query pattern is 5 by default. Both Single-track and All-track Retrieval tasks are executed 50 times and the average time cost is recorded and analyzed.

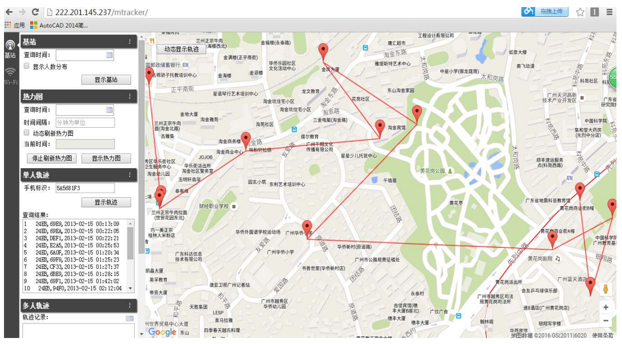
Fig. 3 compares the time cost of trajectory retrieval on different scales of data sets. The Single-track Retrieval is much faster than All-track and the average time cost is about 7 ms. This verifies the high performance of the simple key based search on HBase. On the other hand, the average time of All-track Retrieval on the data set C_7, which contains 104 million records, is 352 ms. The All-track Retrieval is much slower than the Single-track Retrieval due to its computation complexity of matching multiple Spatio-temporal Points. Fig. 4 further shows how the performance of the All-track Retrieval is affected by the number of Spatio-temporal Points in query pattern on the data set C_7. The query time increases proportional to the number of spatio-temporal points.

C. *Prototype system*

Beyond the basic evaluation of the performance of TREST, we have developed a distributed version of the criminal tracing system [2] based on TREST. The system is design to identify the mobile phone of a criminal by retrieving the mobile trajectories, which are matched with a specific moving pattern including multiple Spatio-temporal Points. This is a typical All-track Retrieval task. The system also offers a user friendly web-based UI to visualize search results, which are displayed on the Google Map as illustrated in Fig. 5. The detailed background of the system is presented in our preceding research [2].

Table IV
EXPERIMENT DATA SETS

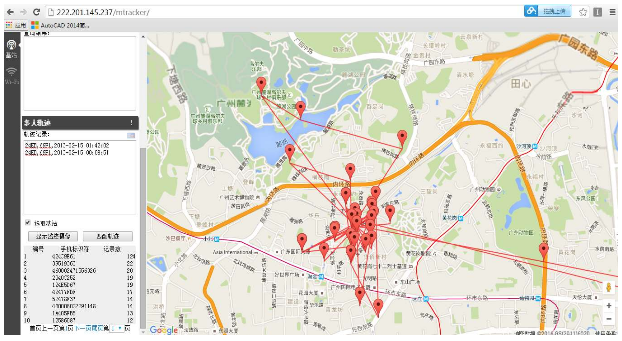
Data set	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
Time range	0am ∼ 1am	1am ∼ 2am	2am ∼ 3am	3am ∼ 4am	4am ∼ 5am	5am ∼ 6am	6am ∼ 7am	7am ∼ 8am
Size	$864M$	$753M$	$684M$	$617M$	$599M$	$624M$	$682M$	$849M$
Data set	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
Time range	0am ∼ 1am	0am ∼ 2am	0am ∼ 3am	0am ∼ 4am	0am ∼ 5am	0am ∼ 6am	0am ∼ 7am	0am ∼ 8am
Size	$864M$	$1617M$	$2301M$	$2918M$	$3517M$	$4141M$	$4823M$	$5672M$



手机标识: 5A5681F3

```
1   24EB,69E9,2013-02-15 00:13:09
2   24ED,69EA,2013-02-15 00:22:05
3   24EB,DEF1,2013-02-15 00:22:21
4   24ED,E2A5,2013-02-15 00:25:53
5   24ED,6A0F,2013-02-15 01:20:34
6   24EB,69F9,2013-02-15 01:25:23
7   24EB,CF33,2013-02-15 01:27:37
8   24EB,6BE9,2013-02-15 01:28:15
9   24EB,69F1,2013-02-15 01:42:02
10   24EB,94F0,2013-02-15 02:12:04
```

(a)



```
24EB,69F1,2013-02-15 01:42:02
24EB,69F1,2013-02-15 00:08:51
```

编号	手机标识符	记录数
1	424C9E61	124
2	39519363	22
3	460002471556326	20
4	2040C252	19
5	124E5D67	19
6	42477FDF	17
7	52478F37	14
8	460008022291148	14
9	1A405FB5	13
10	12586087	12

(b)

Figure 5. The web based UI of the prototype system. a)Single-track Retrieval. b) All-track Retrieval.

VI. Conclusions

In this paper, we present a Hadoop/Hbased based trajectory retrieval system TREST, which is design to support efficient Single-track and All-track retrieval tasks on trajectory data with big and increasing size. TREST directly maps the trajectories intop the simple key-value schema of the HBase, which has a lightweight index structure to support high performance key-based search. Both of the single-tracks and All-track retrieval tasks on the data set with 104 million records can be finished within milliseconds. The throughput of the incremental data insertion on TREST can reached 1.9M/s on a small Hadoop cluster consisting of three normal PCs. The prototype based on TREST, which is designed to identify criminals' mobile phones, also further proves the effectiveness and efficiency of TREST in trajectory retrieval.

Acknowledgments

The work described in this paper was supported by the grants from CAS-NDST Lab, ICT,CAS,100190,Beijing, China(CASNDST201401), the grants from National Natural Science Foundation of China (61300221), and National Training Program of Innovation and Entrepreneurship for Undergraduates (201410561079).

References

[1] Yufei Tao, Dimitris Papadias, and Jimeng Sun. The tpr*-tree: an optimized spatio-temporal access method for predictive queries. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 790–801. VLDB Endowment, 2003.

[2] Jianming Lv, Haibiao Lin, Can Yang, Zhiwen Yu, Yinghong Chen, and Miaoyi Deng. Identify and trace criminal suspects in the crowd aided by fast trajectories retrieval. In *Database Systems for Advanced Applications*, pages 16–30. Springer, 2014.

[3] Kyriakos Mouratidis, Dimitris Papadias, and Marios Hadjieleftheriou. Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 634–645. ACM, 2005.

[4] Philippe Cudre-Mauroux, Eugene Wu, and Samuel Madden. Trajstore: An adaptive storage system for very large trajectory data sets. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 109–120. IEEE, 2010.

[5] Viorica Botea, Daniel Mallett, Mario A Nascimento, and Jörg Sander. Pist: an efficient and practical indexing technique for historical spatio-temporal point data. *GeoInformatica*, 12(2):143–168, 2008.

[6] Haoyu Tan, Wuman Luo, and Lionel M Ni. Clost: a hadoop-based storage system for big spatio-temporal data analytics. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2139–2143. ACM, 2012.

[7] Mingxuan Yuan, Ke Deng, Jia Zeng, Yanhua Li, Bing Ni, Xiuqiang He, Fei Wang, Wenyuan Dai, and Qiang Yang. Oceanst: a distributed analytic system for large-scale spatiotemporal mobile broadband data. *Proceedings of the VLDB Endowment*, 7(13):1561–1564, 2014.