Summary of TrajStore: An Adaptive Storage System for Very Large Trajectory Data Sets

Your name, ID
youremail@fsu.edu

1 INTRODUCTION

In the area of urban computing, many models are based on the trajectory data. Many of them need an efficient operation of the trajectory, such as the real-time map-matched trajectory information. How to query the information from a large scale of trajectory is a key problem in this area. This paper aim to provide a trajectory data management system with high efficient based on cloud computing environment.

2 EXISTING APPROACHES

Nowadays, some cloud computing platform, like Microsoft Azure, can deal with massive data, but they do not perform good for the spatiotemporal query trajectories. Some projects aim to solve this problem, such as Hadoop-GIS, SpatialHadoop, SpatialSpark, and GeoSpark. However, they just work efficient on static data, like road segment and POI. There is no work on providing an efficient way to mine or query trajectory information in a large scale of trajectory dataset.

3 SYSTEM OVERVIEW

To provide a suitable method for query trajectory information in a high efficient from massive trajectory dataset, this paper propose a trajectory data management system based on cloud computing platform. The system contains three components, including Trajectory Storage, Trajectory Spatio-temporal Indexing and Trajectory Map-matching.

3.1 Trajectory Storage

Instead of using Azure blob, the system use Azure Table and Redis server to store the trajectory data. In this component, the system will first group the GPS points based on the trip IDs, order the record by timestamps and rule out the error data. Then store the trajectory to Azure table. In each table, the trajectory data will be divide by temporal range into different partition. Each entry in the table is a record of GPS point, where the partition key is the temporal partition and the rowkey is the exact timestamp. For the ID-Temporal Query Processing, if the query ask for recent trajectory, the system will retrieve data from redis server. If the query ask for historical trajectory data, the system will search from table.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3.2 Inline (In-text) Equations

In this step, the system get trajectory data from redis server and put them into table. For each table, the system will build a fixed spatial index on the trajectory data by equal-sized grids. The GPS points in the same spatial area will be grouped into same partition. For the trajectory in each spatial partition, they will be divide by timestamps. Finally, each entry in the table is a GPS record. The partitionkey is the temporal range, and the rowkey is the combination of timestamp and the trip ID. When querying a trajectory, the system will first check the spatial range of the query and then the temporal range.

3.3 Trajectory Map-matching

The system use Storm to build a system for trajectory map-matching. It includes four parts, Spout, Preprocessing Bolt, Map-Matching Bolt and Report Bolt. The spout get raw data from redis server and send to the preprocessing bolt. Preprocessing bolt remove the error points and send the data to the map-matching bolt. The map-matching bolt map GPS point to road segment. Finally, the report bolt report the result to redis server or Azure table.

4 EVALUATION

4.1 Trajectory Storage

The paper first compared the response time of query for Azure Blob and Azure Table and show table is better than blob. Then it compared the insert time of different partition size. As the side increasing, the insert time decreased. Finally, it compared the response time of different temporal range of query. The smaller the range is, the lower the response time is.

4.2 Trajectory ST-Indexing

The paper first evaluate the insert time in a creasing side of grid size. As the size increasing, the insert time decreased. For the partition Key, the result is as same as grid size, the insert time was decreasing when the key increased. Then this paper evaluate different time range and spatial range of query in increasing grid size. The curves in the two evaluation both first decreased and then increased.

4.3 Storm-based Map-Matching

The paper first evaluated the parallelism degree by using different bolt and datanodes. From the result, we can see that the more datanode were used, the lower response time was. For the number of bolt, the more the better when the number was less than the number of cores in the CPU. In addition, this paper also evaluation the perform of different data node in different number of trajectories, different length of trajectories and different trajectory sample rates. For the number and length of trajectories, the response time were

increasing when the number and length increased. For the sample rate, whee response time were decreasing when the rate increased.

5 STRENGTHS

The paper propose a efficient trajectory manage system for the query of trajectory information and evaluate the performance based on the dataset of real taxi data from Guiyang City, China. To show its advantage, this paper provide 3 practical projects, which are Real-time Taxi Data Management System, Real-Time Traffic Modeling System and Trajectory-based Resource Allocation. It also provide some skills about store and retrieve trajectory data.

6 WEAKNESS

This paper did not compare the performance of its system with other projects. In addition, for the map-matching bolt, it used an interactive-voting based map matching algorithm to map trajectory. However, from my practice before, this algorithm has a large running time. Besides, the system will store the trajectory data in three components duplicately, which wasted massive store space.

7 YOUR SOLUTION

Please provide your solution here. How can you improve this system and address its limitations. XXXXXX