ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 2, February 2015

Smart Travel Guide for Android-Enabled Devices

Namita Badekar[1], Tanvi Parakh [2], Pooja Lokhande[3] , Kaveri Shelake[4] , N.A.Dhawas[5]

Department of Information Technology, Sinhgad Institute of Technology, Lonavala, Maharashtra, India [1,2,3,4,5]

Savitribai Phule Pune University

Abstract: This is an era of smartphones. Nowadays Smartphones usage has become quite popular among the masses. There are quite a lot of android applications available in the market almost in every field. So this paper intents to develop an android app that acts as a smart tourist guide for the users. In Traditional methods, a problem is seen that the travellers do not get exact information while travelling and also the information is not provided on time. The app overcome this drawback by providing convenience to the user while travelling. The app serves following purposes: User can find a companion to share the vehicle(car-pooling),share GPS location and track location of person in friendlist, decide between paths suggested by the application on basis of time, distance, find help in case of any emergency, help the user when he/she is in unknown region ,maintain history of travelling.The app aims to provide detailed Maps and GPS location tracking so users can understand better and can take proper decisions. The travellers can communicate with each other by sending notifications of their GPS location through SMS to other users.
The app works based on the principle of Near Neighbor Join algorithm to calculate distance between two nodes and to a given input node finding a set of nearest nodes using a join function.

Keywords: Android, Google Maps, GPS, kNN Join, MapReduce , Near Neighbor Join.

I. INTRODUCTION

The Smart Travel Guide is an android app that is accessible on Smartphones and Tablet PCs. On installing the app, the user has to register himself by creating his account and maintaining a friendlist by sending/accepting friend request from other users of the app. This app makes use of Google Maps and GPS technology to provide the travellers a detailed Map where the user can search a location, provide a destination and track the route to his destination.

The app serves following purposes:

The user can find a companion to share the vehicle while travelling. For Example, Consider a person John travelling from place A to place B by his Car and needs a companion, in this case the app sends a notification to all the nearest users in John’s friendlist who are willing to travel to the same destination at the same time and thus finding a companion to John to travel. The app on its way to the John’s destination will suggest him which path he should take if there are multiple paths that lead to his destination, by comparing following attributes the minimum distance and time required to reach destination, weather conditions at the destination.

Suppose the traveller is stuck in an Unknown region and is not familiar to that place, then the app helps the user by showing pop ups like nearby restaurant, hotels, hospitals, police stations, fuel pump, etc to help him explore the place.

Suppose the traveller is in an emergency for example, the traveller needs some medical help, so the emergency help finder feature of the app will send notifications to the nearest or closest user(person) informing him about the user’s GPS location and the kind of emergency he is in. If acknowledgement is not received from the nearest user within few minutes(say 3 minutes) then the app searches for the next nearest user, this process goes on until the confirmation of help is received from the nearest user. The app will also send notifications to the nearby hospital asking to provide emergency help.

The other features of the app includes weather forecasting of search and destination location, maintaining travelling history, sharing and tracking GPS location of users in the friendlist, giving feedback(comments, likes)and ratings.

II. LITERATURE SURVEY

The app makes use of following algorithms:

A. Near Neighbor

The traditional approach called Nearest Neighbor Join(also called similarity join), whose goal is to find, based on a given join function, the closest set of objects or all the objects within a distance threshold value to each object in the input. In particular, the app uses a super-scalable system called SAJ-Scalable Approximate Join that is capable of best-effort joining of billions of objects for complex functions[1].

More specifically, SAJ aims to solve the following problem: Given (1) a set I of N objects of type T, where N can be billions; (2) a complex join function F J: I*I->R that takes two objects in I and returns their similarity; and (3) resource constraints.The two key resource constraints are :1)the number of objects each machine can be expected to perform an all-pairs comparison on and 2)the maximum number of records each Shuffle phase in the program is able to handle, which can be derived from the number of machines available in the cluster. For all o to I, find k objects in I that are similar too according to FJ without violating the machine constraints[1].

At a high level, SAJ operates in two distinct phases. In the Bottom-Up (BU) phase, the set of input objects are iteratively partitioned and clustered within each partition to produce a successively smaller set of representatives.

Each representative is associated with a set of objects that are similar to it within the partitions in the previous iteration. In the Top-Down (TD) phase, at each iteration the most similar pairs of representatives are selected to guide the comparison of objects they represent in the upcoming iteration[1].

B. MapReduce

A MapReduce may be a programming model that does the processing of huge amounts of structured and unstructured data in a vast cluster in an exceedingly reliable and fault tolerant manner.

A MapReduce splits huge set of data into freelance chunks and organizes the input file as a key-value pairs. Map operator takes an input key-value pair and produces a collection of intermediate key-value pairs. MapReduce runtime system then teams and sorts all the intermediate values related to a similar intermediate key, and sends them to the Reduce function. Reduce function accepts an intermediate key and its corresponding values, applies the process logic, and produces the ultimate result.

III. PROPOSED SYSTEM

However comparing billions of objects with each other to find the set of nearest objects to the input object(as described in Near Neighbor Join Algorithm) is very time consuming which involves lots of processing.

To overcome this drawback we use a technique called k-Nearest Neighbor Join(kNN) using MapReduce, a mapping mechanism that does distance filtering, and therefore reduces both the shuffling and the computation costs.

A. System Architecture

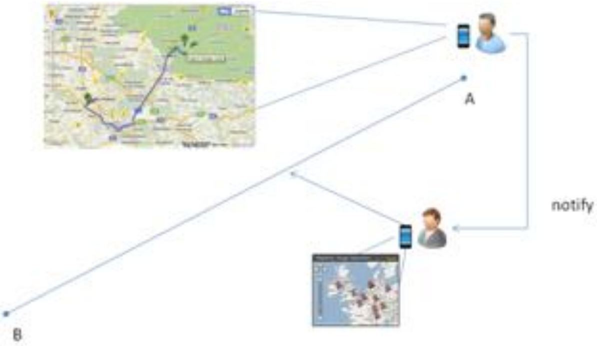

Fig.1 System Architecture

The system architecture consists of the user who specifies its destination location and the application displays the GPS location of all the other users travelling to that location (destination).A travelling route of the user is defined from source to the destination. The other users are notified of the current GPS position of the user.

B. System Flow

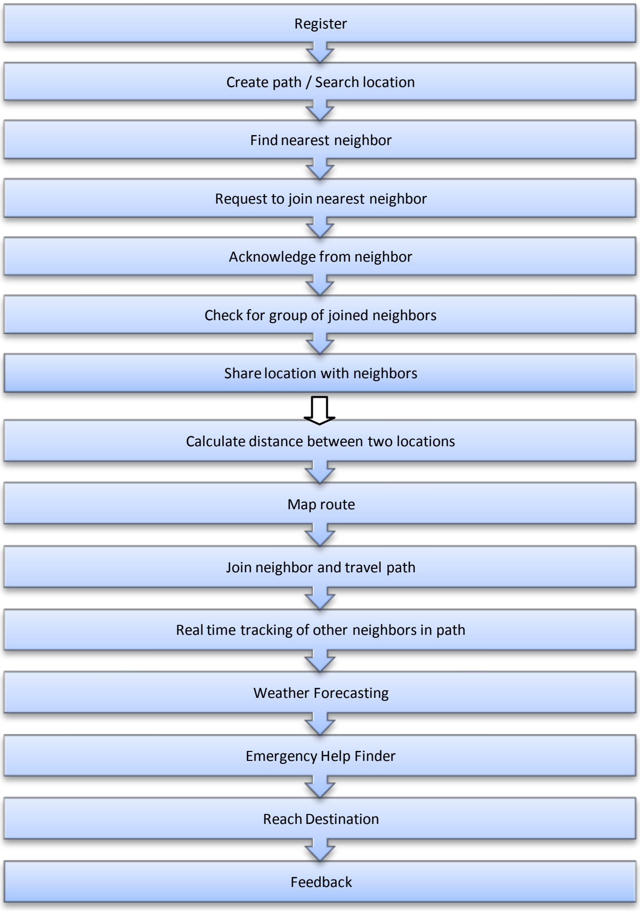
Register
Create path / Search location
Find nearest neighbor
Request to join nearest neighbor
Acknowledge from neighbor
Check for group of joined neighbors
Share location with neighbors
Calculate distance between two locations
Map route
Join neighbor and travel path
Real time tracking of other neighbors in path
Weather Forecasting
Emergency Help Finder
Reach Destination
Feedback

Fig.2 System Flow

C. Algorithm

1) Near Neighbor

Select * from Travel_master where
Travel_Destination=”user_dest” and
travel_Visibility =”public ”or ”private”or ”select”

2) MapReduce

```
\\input
Select all from tbl_Users
\\selective input
 Select data from Friends where userid=”user_id”
 Select threshold from Preference
 \\refresh list
 Update location_user from User_Tracker
 \\partition
 Select top(n) from Friends join location_user
 where userid=”user_id” and threshold <
 Geography(location_user) and
 TotalDistance(location_friends)
```

IV. FUTURE APPLICATIONS

- This application can be used as smart system which will be more sophisticatedly working for benefit of the user. A user can be made aware about surroundings even in unknown region to him/her.
- This application can be used as a smart emergency help application due to algorithm used. The

DOI 10.17148/IJARCCE.2015.4270

algorithm used has capacity to notify the emergency to user’s nearby contact, instead of notifying to every or specifically selected contacts.

- This application can be used as pooling app i.e. a companion can be found out while travelling from one place to another.

V. CONCLUSION

- Thus the Android application being developed will overcome the difficulties faced during travelling and will help user to remain in contact with other users.
- The application will constantly provide the user with surrounding details and information, making the journey of the user convenient.

REFERENCES

[1] Herald Kllapi, Boulos Harb and Cong Yu, “Near Neighbor Join”, Univ. of Athens ,Greece, 978-1-4799-2555 Jan 14, 2014

[2] W. Lu, Y. Shen, S. Chen, and B. C. Ooi, “Efficient processing of k nearest neighbor joins using MapReduce,” PVLDB, vol. 5, no. 0,2012.

[3] A. Okcan and M. Riedewald, “Processing theta-joins using MapReduce,” in SIGMOD, 2011.

[4] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” Commun. ACM, vol. 51, pp. 107–113,January2008.[Online].Available:http://doi.acm.org/10.1145/1327452.1327492

[5] C. B¨ohm and F. Krebs, “High performance data mining using The Nearest neighbor join,” in ICDM, 2002, pp. 43–50.

[6] H. Lee, R. T. Ng, and K. Shim, “Similarity join size estimation using locality sensitive hashing,” PVLDB, vol. 4, no. 6, pp. 338–349, 2011.

[7] C. Yu, R. Zhang, Y. Huang, and H. Xiong, “High-dimensional knn Join with incremental updates,” GeoInformatica, vol. 14, no. 1, pp.55–82,2010.

[8] J. C. Shafer and R. Agrawal, “Parallel algorithms for high-Dimensional similarity joins for data mining applications,” in VLDB, 1997.

[9] B. Yao, F. Li, and P. Kumar, “K nearest neighbor queries and knn-joins in large relational databases (almost) for free,” in ICDE, 2010, pp. 4–15.

[10] Y. Chen and J. M. Patel, “Efficient evaluation of all-nearest-Neighbour queries,” in ICDE, 2007

[11] C. Xiao, W. Wang, X. Lin, and H. Shang, “Top-k set similarity joins,” in ICDE, 2009

DOI 10.17148/IJARCCE.2015.4270