Finding Nearest Facility Location with Open Box Query using Geohashing and MapReduce

Vikram Singla
Computer Science & Engineering Department
Thapar University
Patiala, India
vsingla1989@gmail.com

Dr. Deepak Garg
Computer Science & Engineering Department
Thapar University
Patiala, India
dgarg@thapar.edu

Abstract—**This paper gives an algorithm to find the nearest facility location using Geohashing. Open box query is implemented to find nearest location which is based on unbounded data. Searching is done based on the longitude and latitude of locations. Geohashing is a technique which converts the longitude, latitude pair into a single value which is represented in binary format. Data accumulated in Geospatial queries is too big in size. Therefore MapReduce framework is used for parallel implementation. MapReduce splits the input into the independent chunks and execute them in parallel over different mappers. Geohashing and MapReduce when fused together to find the nearest facility location gives very good results.**

Keywords—Geospatial, Geohashing, Point of Interest, Hadoop, MapReduce.

I. INTRODUCTION

This paper presents the algorithm to find the nearest location (point of interest) around any current location. Those locations are point of interest. The location can be of any category like school, library, hotel, temple and so on. Main concern here is to draw a more accurate and efficient solution to this problem. It is desired that one could query the unbounded data means there is no limitation on the range of data. There are mainly two types of queries, one that solves bounded box problem that involves limited data and other are unbounded box or open box which queries wide range of data [1]. Social data, now-a-days, is accumulated in large sizes [2].

So as this algorithm will find the nearest location there will be no limitation like one can find it in the range of 10 km only or anything like that. Anyone would be able to find the nearest location over a whole country or more than that. So we here provide the solution for that problem using MapReduce [3]. In MapReduce execution takes place in parallel [4], so also it will take less time while executing on the large datasets. MapReduce is mainly useful in those cases where large amount of data is involved. Many methods are already there to find nearest location, but somewhere all of them have some or more flaws.

II. MAPREDUCE

MapReduce is a parallel programming technique which is used for processing very large amounts of data. Processing of this much large amount of data can be done efficiently only if it is done in a parallel way. Each machine handles a small part of the data. MapReduce is a programming model that allows the user to concentrate more on code writing. Coders need not be worried about the concepts of parallel programming like how to distribute the data on different machines or what will happen on the failure of any machine etc. This all is inbuilt in MapReduce framework. In MapReduce, the input work is divided into various independent chunks. These chunks are processed by the mappers in the totally parallel way. The mappers process the chunks and produce the output and then this output is sorted and fed as input to the reducers. The framework itself will handle scheduling, monitoring and re-execution of the tasks which have been failed. MapReduce allows a very high aggregate bandwidth across the cluster because typically the nodes that store and the nodes that compute are the same.

The MapReduce framework accepts the input as a set of <key, value> pairs and produces the output also in a set of <key, value> pairs, though they might be of different type. The inputs and outputs to the MapReduce framework can be shown as:

map (k1,v1)	→	**list(k2,v2)**
reduce (k2,list(v2))	→	**list(v2)**

First line represents the map function which shows that map function took the value in the form of key value pairs which is represented <k1,v1>. Map function after processing those key value pairs, produces a new set of key value pairs. This time these key value pairs can be of different type, therefore they are represented by <k2,v2>. This set of key/value pair that is <k2,v2> is fed to the reducer.

III. PROPOSED WORK

Two functions Map and Reduce have been defined to carry out all the work. MapReduce framework works on these two functions. . The framework itself will handle scheduling of tasks, monitoring of tasks and re-executing the tasks which have been failed [5]. Geohashing [6] algorithm has been used in Map function. Output given by the Map function is taken as input by the Reduce function.

The map function receives the <key, value> pairs in the form of <longitude:latitude, type:place>. Longitude and latitude are given in the decimal format. These (longitude, latitude) combinations are converted into the binary format

978-1-4799-2572-8/14/$31.00 ©2014 IEEE

using the Geohash function. The longitude and latitude of the current location is taken from where nearest location has to be found. After receiving that (longitude, latitude), this combination is converted convert into bitwise fashion using the geohash function and save it in variable geocode. Now the geohash location index of the current location has been obtained. Specifically, each of Geohash pair can be created by the mappers in parallel. Now the same is done for all the locations that are there in the database so that they all can be compared with the current location. Various categories of locations could be present like ATM, Bar, Bus station, Café, Fuel, Library, Pub and Restaurant. Prefix matching of each line is done with the input value. It can be seen that up to what prefix length, values match. If prefix matches to the full length then it means that a very high precision has been obtained and one is standing almost near to the destination location for which he is searching. So the map function will provide output as (prefix_matched, place_type, place_name). And this output will be given as input to the Reducer. Procedure of the mapper can be learnt more from the following flowchart which shows step to step implementation of mapper.

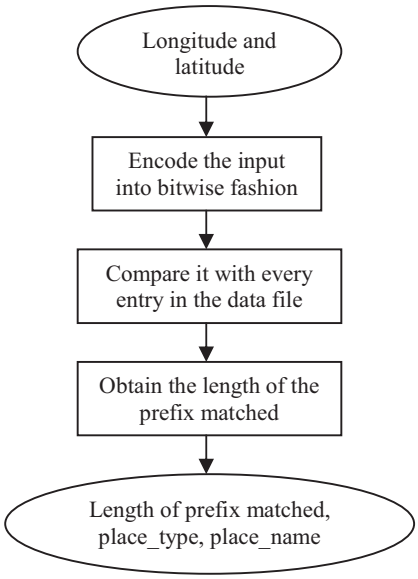


Fig. 1. Flowchart showing map task

Algorithm 1 Map

```
Procedure Map (input_lon, input_lat ,place_type)
1.  geocode ← geohash(input_lon, input_lat)
2.  for line in datafile do
3.      // Remove leading and trailing whitespace
4.      line ← line.strip()
5.      // Split the line into words
6.      data ← line.split(',')
7.      if len(data) = 4 then
8.          lat ← float(data[0].strip())
9.          lon ← float(data[1].strip())
10.         geo ← geohash (lat,lon)
11.         length_code ← len(geo)
12.         count ← 0
13.         for long in xrange (0,length_code) do
14.             if geocode.startwith
                (geo[0:length_code - count]) then
15.                 break
16.             else
17.                 count ← count+1
Output Length of prefix matched, Place type, Place name
```

First of all, longitude and latitude of the current location is provided from where the nearest location has to be found, then it is encoded by geohash function in a geocode, and same is done to all locations present in data file and then compared to the geocode of current location. After that prefix matched length is obtained for each entry in the data file.

Algorithm 2 Reduce

```
Procedure Reduce(prefix_length, place_type, place_name)
1.  type ← place_type
2.  for line in datafile do
3.  // Remove leading and trailing white spaces
4.      line ← line.strip()
5.      data ← line.split()
6.      if type = "all" then
7.          Display all the place names in the order
                from nearest to far
8.      else if type = data[1].split(':') then
9.          Display all the place names of the specified
                type in the order from nearest to far
```

The reduce function will accept prefix_length , place type and place name as the input. It no longer requires the geohash code because only names will be needed now to display them. Mainly filtering is being done in the reduce function. Nearest location may be required to found out of every category altogether or it may be required to found out of a particular category. So, based on that criterion, filtering is done. It means filtering is done based on the type or category of place to be found nearest to the current location. When the output of the map function is provided as input to the reduce then before that all the <key,value> pairs are sorted automatically by the Hadoop[7]. No separate sorting is required to be done in the map function. MapReduce factors can be tuned very carefully, according to the need [8].

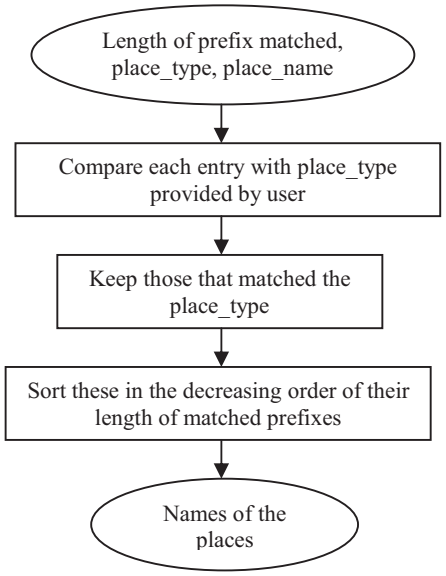


Fig. 2. Flowchart showing reduce task

Entries corresponding to every category are passed to the reduce function. Now the reduce function will match the type of these entries one by one with the type provided by the user. If the type of the entry will match with the type provided by the user then it will be displayed. User may be interested in any location of his choice, like may be interested in finding the nearest restaurants where she want to go or she might be interested in finding temples. So reduce function will filter that based upon that. The data is already in sorted order because Hadoop did it internally. They are sorted in the order of their length of prefix matched. So the entry whose length of prefix match is longest means it is nearest to the current location where user is standing.

Algorithm 3 Geohashing

Procedure Geohashing(longitude , latitude)

1. First of all the world is divided into two halves. The first half is given value "0" and the another half is given the value "1". The division is done with a vertical line.
2. Each division is again divided into 2 parts. Division is done with a horizontal line. Each divided portion is again assigned the numbers "0" and "1". "0" is assigned to the one half and "1" is assigned to the another half.
3. This is done until a very long series of bits is obtained.
4. This bitwise series is converted into the base-32 format.
5. In whichever region the location is falling, the code of that region is the geohash code of the location.

Geocode has a feature that as the number of bits decreases, the precision to find the location of a place also decreases. Due to this property of the geocode, this geocode can be used for nearest location search. Geocode helps in giving the nearest locations as the location which will be near will have similar geohash prefixes. If the prefix is matched to another prefix for complete length, it means they are at same location or very near to each other. Therefore this geocode is compared with the geocode of all other places whose record is there. It will tell that which is nearest amongst all those places. Geohash algorithm uses a base-32 numbering system.

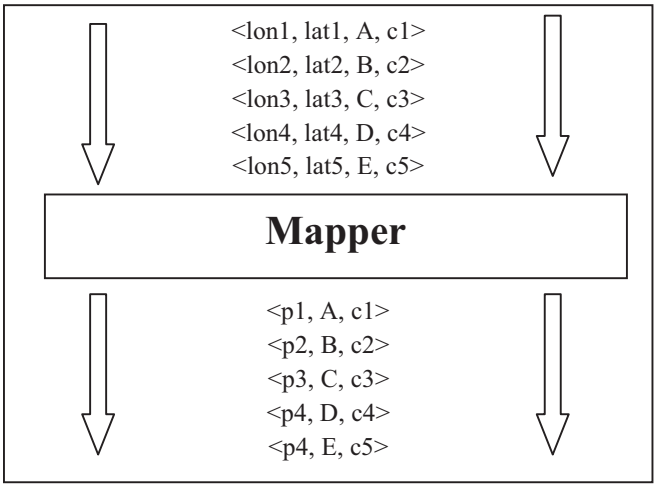
IV. Explanation with Example

Let the longitude and latitude of the location from where any nearest location has to be found be **<lat, lon>**. "lon" here represents the longitude and "lat" represents the latitude of a current location.

There would be a data file which contains the information about all the locations. Each information set about any location contains 4 values, **latitude of location, longitude of location , name of location and type or category of location**. Here an example is shown by taking a current location and five more locations present in the data file with their set of information as

<lat1, lon1, A, c1>
<lat2, lon2, B, c2>
<lat3, lon3, C, c3>
<lat4, lon4, D, c4>
<lat5, lon5, E, c5>

Here **Latitude = {lat1, lat2, lat3, lat4, lat5}** depicts the latitude of the corresponding location. **Longitude = {lon1, lon2, lon3, lon4, lon5}** depicts longitude of the corresponding location. **A, B, C, D, E** are the names of locations and **categories of location = {c1, c2, c3, c4, c5}** where any two categories can be of same type. Just to show five different entries we here gave the categories five different names. Now mapper and reducer will do the following work.



The mapper receives the information about the locations in the form of <key, value> pairs. So here the <key,value> pair is **<longitude:latitude, place_name:place_type>**. These longitude and latitude values are converted into the

corresponding geohash code for each of the five entries. And same is done for the current location.

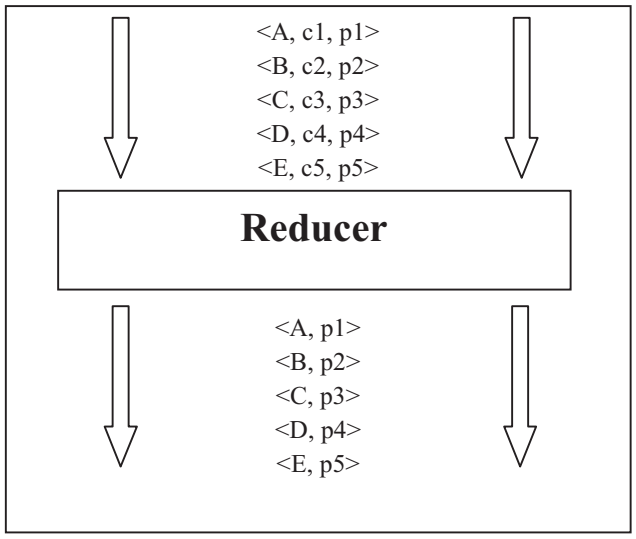


Fig. 4. Inputs and outputs to reducer

Then each of the geohash code of the five entries is matched with the geohash code of the current location to find that up to what length the geohash codes of five locations match to that of current location. **Length of prefixed matched = {p1, p2, p3, p4, p5}** means these are the lengths of the prefix matched for each location to that of current location. This helps in forming the new <key,value> pair which is given as output of the mapper. So the new <key,value> pair is **<length_of_prefix_matched, place_name:place_type>.** This new pair is fed as input to the reducer.

The main task of the reducer is to do the filtering. Reducer now knows that which locations are nearest and which are far based on the length of prefix matched of each location. Reducer now does the filtering based on the type of location provided by the user in which he is interested. Reducer will compare each entry based on type or category of location with type provided by the user and would display only those names which match the type.

IV. Conclusion and Future Scope

The amount of geospatial data is rapidly growing. With the advancement in web technology and availability of spatial data infrastructure, the demand for accessing geospatial information over web has increased significantly. Geospatial queries play an important role in a wide variety of applications such as decision support systems, profile-based marketing, bioinformatics and GIS. In this paper a method to tackle nearest location problem has been derived using Geohashing. The size of the data in this becomes too big. It would be inefficient to access that data sequentially. So the MapReduce has been used to execute the data in parallel. It improves the solution in terms of time complexity to a great extent, because the data is being processed in parallel. A variety of geospatial queries can be efficiently modeled using MapReduce. MapReduce is a programming model that lets developers focus on writing the code that processes their data without having to worry about the details of parallel execution. A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. The MapReduce framework is very well suited for the Geospatial queries.

Geospatial queries play an important role in modern day as geospatial data [9] is growing rapidly. And finding the nearest location is one of the most important geospatial queries. Finding the nearest location helps in decision making for various organizations. Like a telecom industry may be interested in finding some nearest location where it can implant its towers and by many other companies in some way. It can be helpful and used in day to day life like when anyone has to find any nearest location around him which he wants to visit or want to have knowledge about. It would be interesting to see if nearest location method is be fused with any other method to help in better way for decision making. Then the tradeoff would be done between distance and other factor that would be fused. It would be of great help for various organizations. Like for a telecom industry if it wants to find the nearest location around it where it can start its service but at same time has adequate population and similarly for other organizations.

References

[1] P. Hitzler and K. Janowicz, “Linked Data, Big Data, and the 4th Paradigm”, IOS Press and the authors, Semantic Web 4, 2013.

[2] W. Fan, “Querying Big Social Data”, Lecture Notes in Computer Science, Volume 7968, 2013, Springer Berlin Heidelberg, BNCOD 2013, pp 14-28.

[3] J. Dean, S. Ghemawat, “MapReduce: simplified data processing on large clusters.”, Communications of the ACM - 50th anniversary issue: 1958 – 2008, Volume 51 Issue 1, January 2008, Pages 107-113.

[4] W. Pan, Z. Li,Q. Chen,S. Peng,B. Suo,J. Xu, “MSMapper: An Adaptive Split Assignment Scheme for MapReduce”, Lecture Notes in Computer Science, Volume 7419, 2012, Springer Berlin Heidelberg, 2012, pp 162-172.

[5] H. Jin, K. Qiao, X. Sun, Y. Li, “Performance under Failures of MapReduce Applications”, IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) 2011, pp. 608-609.

[6] “Geohash and its format”, [online] Available: http://www.geohash.org/site/tips.html(Accessed: 2013, September 5)

[7] “What is hadoop” [online] Available:http://www.hadoop.apache.org (Accessed: 2013, September 5)

[8] W. Premchaiswadi, W. Romsaiyud, “Optimizing and Tuning MapReduce Jobs to Improve the Large-Scale Data Analysis Process.”, International Journal of Intelligent Systems, Vol. 28 No. 2, 2013, pp. 185-200.

[9] A. Akdogan, U. Demiryurek, F. B. Kashani, C. Shahabi, “Voronoi-Based Geospatial Query Processing with MapReduce”, IEEE Second International Conference on Cloud Computing Technology and Science, 2010,pp. 9-16.