Finding Nearest Location with Open Box Query using Geohashing and MapReduce

Thesis submitted in partial fulfillment of the requirements for the award of degree of

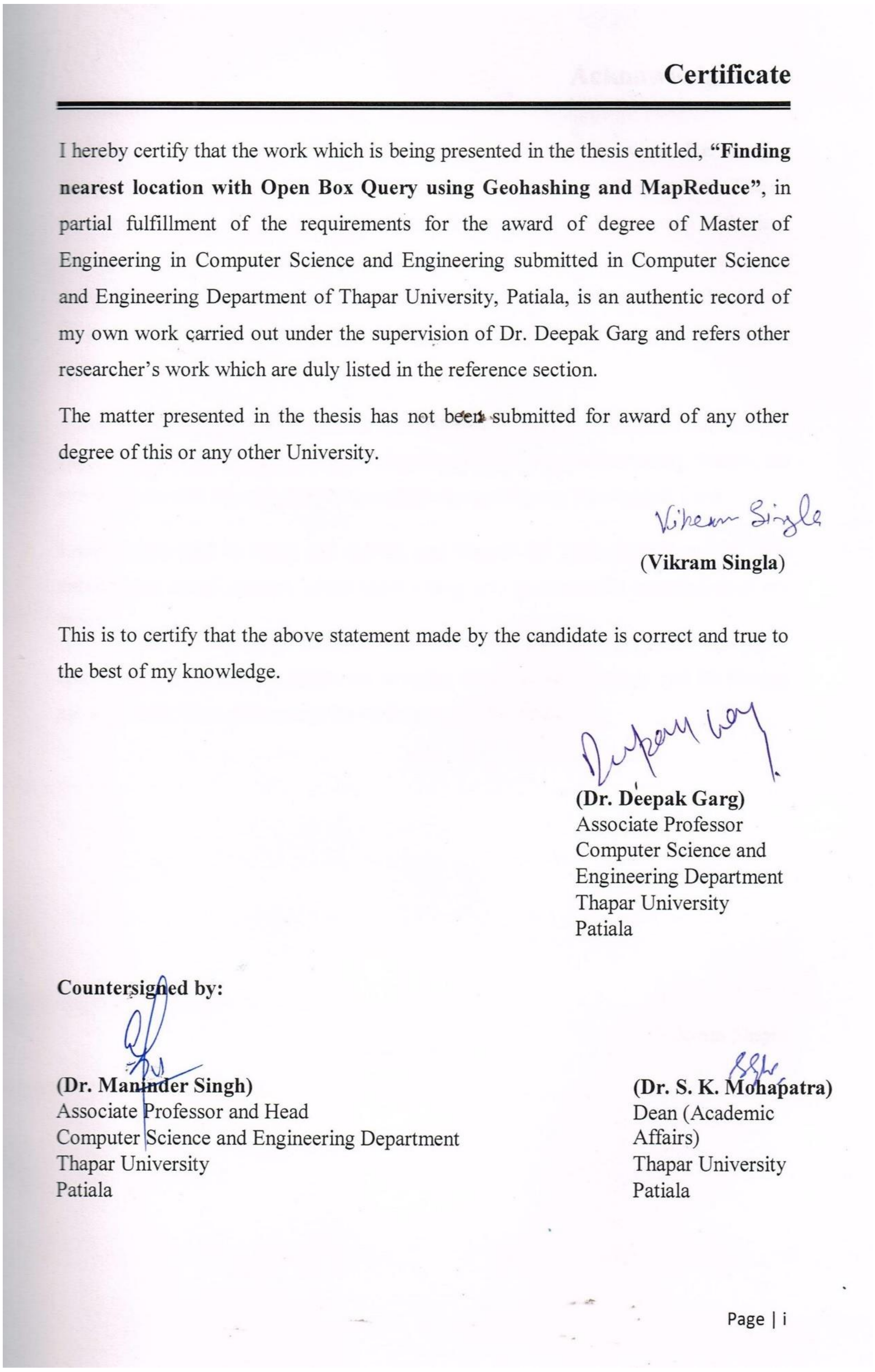
Master of Engineering
in
Computer Science and Engineering

Submitted By
Vikram Singla
(Roll No. 801132029)

Under the supervision of:
Dr. Deepak Garg
Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2013



Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Finding nearest location with Open Box Query using Geohashing and MapReduce"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Deepak Garg and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Vikram Singla

(Vikram Singla)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

[illegible]

(Dr. Deepak Garg)
Associate Professor
Computer Science and
Engineering Department
Thapar University
Patiala

Countersigned by:

[illegible]

(Dr. Maninder Singh)
Associate Professor and Head
Computer Science and Engineering Department
Thapar University
Patiala

[illegible]

(Dr. S. K. Mohapatra)
Dean (Academic
Affairs)
Thapar University
Patiala

Acknowledgement

I would like to express my deep sense of gratitude to my supervisor, **Dr. Deepak Garg**, Associate Professor, Computer Science and Engineering Department, Thapar University, Patiala, for his invaluable help and guidance during the course of thesis. I am highly indebt to him for constantly encouraging me by giving his critics on my work. I am grateful to him for giving me the support and confidence that helped me a lot in carrying out the research work in the present form. And for me, it's an honor to work under him.

I also take the opportunity to thank **Dr. Maninder Singh**, Associate Professor and Head, Computer Science and Engineering Department, Thapar University, Patiala, for providing us with the adequate infrastructure in carrying out the research work.

I would also like to thank my parents and friends for their inspiration and ever encouraging moral support, which went a long way in successful completion of my thesis.

Above all, I would like to thank the almighty God for His blessings and for driving me with faith, hope and courage in the thinnest of the times.

Vikram Singla

Abstract

Geospatial queries play an important role in modern day life both for common man as well as industries. Due to advancement in location based industries this need has grown even more. Geospatial query can be of various types. In this thesis geospatial query to find the nearest location is presented.

Geohashing will be used to come out with the solution. Geohashing is a technique which converts the longitude, latitude pair into a single value. Usage of the geohashing will make it very efficient to find the location. There comes various situation when one want to know the any nearest location around him in which he/she is interested due to any reason.

Geospatial query uses Geospatial data. But the size of this data is too big. So it is inefficient to process this much big data using sequential methods. Therefore some parallel processing technique is required to make it more efficient in various manners.

MapReduce is a framework that is used for parallel implementation. MapReduce splits the input into the independent chunks and execute them in parallel over different mappers. When using MapReduce, developer need not to worry about other factors like fault tolerance, load balancing etc. all these factors are handled by MapReduce. He can only concentrate more on his programming. Without MapReduce it would not be efficient to process so much big spatial data. Spatial data may contain the knowledge about lot of locations. So it is very important to efficiently process it.

MapReduce has been used in various fields before but never has been used in finding the nearest location. Geohashing and MapReduce when fused together would give very good results.

Table of Contents

List of Figures

Chapter 1
Introduction

1.1 Big Data

Big data[1] is a collection of large data sets. Due to their giant size it is not efficient to process those using traditional methods. Various problems that have to be faced in processing big data are capturing the data, storage, search, sharing, transferring, analysis etc. The trend to giant data sets is because of extra information derived from analysis of a single large set of correlated data, as compared to separate smaller sets with the equivalent total amount of data. This big data helps in finding the relations between various fields which may help in various ways, like decision making, understanding the business trends, long term planning, fighting crime, and getting real-time roadway traffic conditions[2]. But due to their correlated behavior it becomes difficult to query them[3]. Professionals are trying to make results from this huge amount of data.

This explosion of data is seen every sector in the computing industry. The Internet companies such as Google, Yahoo, Facebook etc have to deal with large amounts of data generated by the user and this data is in the form of blog posts, photographs, status messages, and audio/video files. Also there is huge quantity of data which is indirectly generated by web sites in the form of access log files, click through events etc. Analysis of this data can induce useful patterns about the behavior of user. Most of this data is generated frequently, and the data sets are stored temporarily for a fixed period and once they are used, they are discarded after that. According to Google, the online data on web today is 281 Exabyte, which was 5 Exabyte in 2002[4]. There has been a tremendous increase in user generated data since 2005.

1.1.1 Uses of big data

- People are sharing more and more data online like sharing their personal lives, their opinions on peoples and products.
- Companies are trying very hard to use this data in order to generate user preferences, to generate better options for users and to just get to know their users more and more.

- Big data is beneficial in various situations like if one wants to have minute to minute or second to second information of anything, then big data will be accumulated having all the records.
- Big data helps in keeping the information about real time traffic conditions, it may help in better understanding them and avoid any jams.
- If minute to minute data about the weather is kept, then it can help in avoiding the situations like floods or any other calamities.
- In capital market , big data is used to study trend of the market[5].

1.2 Geospatial query

Any query related to geospatial data is known as geospatial query. Geospatial data is the data that is related to locations or defines a particular location. It contains the information within it about the geography of locations. Geospatial can be said as data related to land management. Due to recent advancements in services where location is required, the amount of geospatial data has grown very fast. And therefore there is a demand for the geospatial queries which can efficiently process this large amount. Geospatial data is being accessed by already many geospatial frameworks. So geospatial querying involves processing all the geospatial data and find something useful out of it that the user needs. So soon when this data will increase to an even larger size then we would require more efficient and fine methods to process geospatial data to produce something beneficial out of it which would solve a lot of purposes and answer our questions which would help in many matters of day to day life. A variety of geospatial queries can be efficiently modeled using MapReduce[6].
Geospatial data play a very important role in modern day life as it helps in lot of matters of any person like long term planning,

- In disaster management, in emergency situation. For example if there is a earthquake somewhere, then geospatial information is useful in knowing what is nearest location where are victims can be evacuated and in emergency situations like to find nearest hospital and all .
- Due to the importance of this geospatial data, importance of geospatial queries also increases. Geospatial data may be present in various forms and are too

diverse in nature. They may be present as digital images, in tables, vector data or in any other form. Today even some non profitable organizations and volunteered people are updating Geospatial information System (GIS)[7] or geospatial datasets with authoritative information.

- This has really contributed a lot in collecting the data about various geospatial sites. Advancement in sensor technology and software and hardware technology has made the accumulation of this data in large amounts.

1.2.1 Types of geospatial queries

The geospatial queries can be broadly categorize into two major categories.

- First type is bounded box queries where queries is specified for some range r_k to locate point of interest P_k .The complexity of finding P_k exponentially increases with r_k. P_k~$0(r_n^2)$. These types of queries are comparatively easier to solve as bounding boundaries can be determined from r_k. The amount of data involved with the queries vary with the r_k and typically they are applied on small ranges like 10 km.

- The other category of queries are unbounded where P_k is known but r_k is not known at the time of query like (nearest airport, whereby can find snow in the month of july). Since r_k is not known the complexity of finding result of such query will be proportional to data aggregated. This type of problem is tougher to solver involving large set of data involved in it. Geospatial queries can be used in a number of ways, like nearest neighbor queries and reverse nearest neighbor queries.

1.3 Point of interest

A point of interest, or POI, is a specific point location that someone may find useful or interesting. As an example it can be any point on Earth representing something like a hotel, garden, school etc or it can be even a point on Mars which is representing the location of the mountain or any other thing there. Many of the consumers use the term when referring to hotels, campsites, fuel stations or any other categories used in modern (automotive) navigation systems. Like when one want to know about any location where he want to go or want to have knowledge about then that location is known as point of interest. Geospatial queries are used to find point of interests in various ways. One may put a query to find any point of interest that one is interested in. Point of interest is used by many industries for their benefits in a number of ways.

A telecom industry may require knowing its point of interests for gaining information that which locations are most important to have their telecom towers. This point of interest can be based on the various parameters like population, area, height etc. In image processing point of interests are the locations which are of importance like those points which are required for the reconstruction of any image. In GPS the point of interests are the locations that one want to find on the Map which is of our interest due to any reason either one want to know about it or visit, it can be a forest, a city, monument, temple or anything else.

1.4 Geohashing

Geohashing is a way of encoding the latitude and longitude and grouping of the nearby points on the globe with different resolutions. There is a set of coordinates for each 1°×1° latitude longitude zone which is known as graticule in the world. The coordinates of longitude and latitude can lie anywhere in the graticule. It can be in the forest, in a lake, on a city, or anywhere else.

The geohash code is obtained in the following way:

First of all the world is divided into two halves. The first half is given value “0” and another half is given the value “1”. The division is done with a vertical line. Let the location to be searched is in the half “0” of the earth.

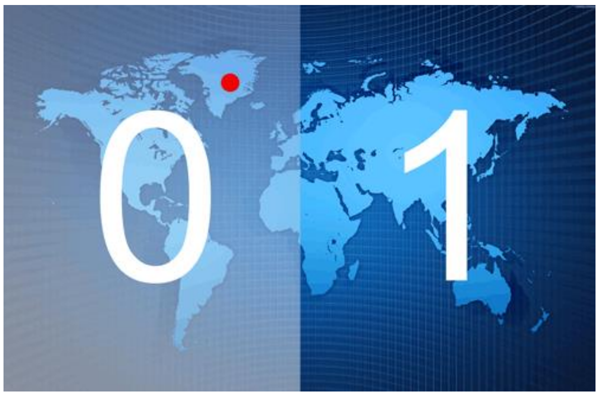


Figure 1.1: Subdividing the earth surface up to one bit [8]

But in this situation one can be helpless as none of his purpose would be solved due to such low precision. It is the case of lowest precision and as well as the geohash code generated is also of lowest precision and resolution.

Now this resolution needs to be improved. So to provide this precision the “0” half of the earth is further divide into two halves. This time the division is done by a horizontal line.

Now again assign “0” to one half and “1” to another half. It is seen that dot is located in the “0” half. So this time the geohash code is more précised and has more resolution that is “00”. One “0” comes from first and second “0” came from second division.

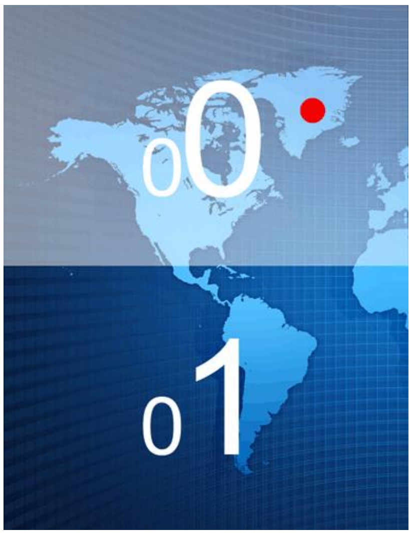


Figure 1.2: Subdividing the earth surface up to two bits [8]

Now in order to improve this precision further, “00” portion of earth is again divided into two halves named “0” and “1”. And this time even more precision is got ,that is “001” as seen in the figure 1.3

In this way each sub-division keeps on dividing vertically and horizontally, means alternately dividing longitude and latitude. This converts the longitude and latitude information to a single value.

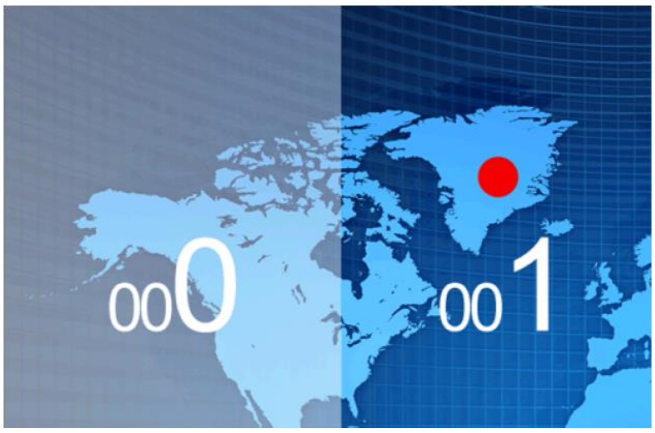


Figure 1.3: Subdividing the earth surface up to three bits [8]

After keep sub dividing in this way very high precision value is get, like up to street level or even beyond that. At that level Geohash will look a bit longer and can be shown as this.

```
0011110001011001011010100011110110110101
```

This binary is represented as alphanumeric characters (32 bit encoded). Each 5 bits are converted to one character.

```
00111  10001  01100  10110  10100  01111  01101  10101
```

So by converting latitude and longitude information in a bitwise fashion, a single value is generated that provides a very high resolution geographic point, and is very well suited for storage or transmission in the form of a character string.

Geohash code has a feature that as the number of bits decreases, the precision of finding the place also degrades. It is obvious that higher the number of bits, high will be the precision[8].

1.5 MapReduce

MapReduce[9] 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[10]. He needs not be worried about the concepts of parallel programming like how he will distribute the data to different machines or what will happen on failure of any machine etc. This all is inbuilt in MapReduce framework.

In MapReduce, the input work is divided into various independent chunks[11]. These chunks are processed by the mappers in the totally parallel way. The mappers process the chunks and produce the output, then this output is sorted and fed as input to the reducers. The framework itself will handle scheduling of tasks, monitoring of tasks and re-executing the tasks which have been failed[12]. Mapreduce allows a very high aggregate bandwidth across the cluster because typically the nodes that store and the nodes that compute are the same. Means the MapReduce framework and distributed file system runs on the same set nodes due which it provides high aggregate bandwidth.

The input to this framework is provided in the form of key/value pairs and it produces the output also in the form of key/value pairs. All the coding is done in the form of two functions that is Map and Reduce. So the developers must have knowledge that how he will represent his coding in the form of Map and Reduce functions[13]. The output of the Map function is sorted and fed to the Reduce function. MapReduce is very suitable when there is a large amount of data involved. For small amount of data it might not be that useful. MapReduce can be run on the commodity hardware with reliability and in a fault tolerant way. Therefore it can be said that MapReduce is a software framework which allows to write the programs with a ease which involves large amount of data and the developer need not to worry about anything else when his data is being processed parallelly.

Before MapReduce large scale data processing was difficult because before that one had to look after the following factors himself but now they all are handled by the framework.

- Managing thousands or hundreds of processors
- I/O Scheduling
- Status and monitoring
- Managing parallelization and distribution
- Fault/crash tolerance

MapReduce provides all of these easily because these all are inbuilt in Map Reduce framework[14]. Before MapReduce one needed to have a complete knowledge about all these factors, had to do separate coding for them like how to distribute the data if any node fails, keeping a check on processors etc. But now with MapReduce all that is inbuilt, user only can concentrate on his own implementation.

1.5.1 Types of nodes in map reduce

- **JobTracker** node manages the MapReduce jobs. There is only one of these per cluster. It receives jobs submitted by the clients. It schedules the map tasks and reduce tasks on the appropriate TaskTrackers in a rack aware manner and moitors for any failing task that need to be rescheduled on a different TaskTracker.
- **TaskTracker** nodes are to achieve parallelism for map and reduce tasks, there are many TaskTrackers per cluster. They perform map and reduce operations.

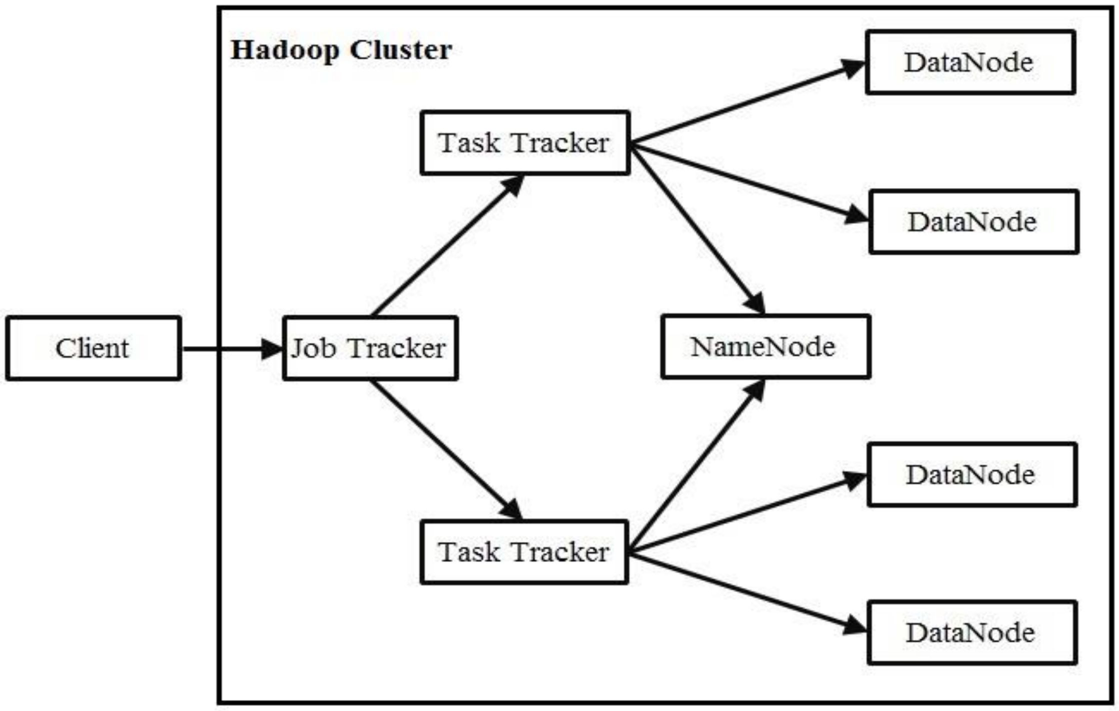


Figure 1.4: Various nodes in MapReduce[15]

There are also NameNode and DataNodes and they are a part of hadoop file syatem. There is only one NameNode per cluster. It manages the the filesystem namespace and metadata. Expensive hardware commodity is used for this node. There are many

DataNodes per cluster, these manages blocks with data and manages them to the clients. They periodically reports to the NameNode the list of blocks it stores. Inexpensive commodity hardware is used for this node.

1.5.2 MapReduce Execution Overview

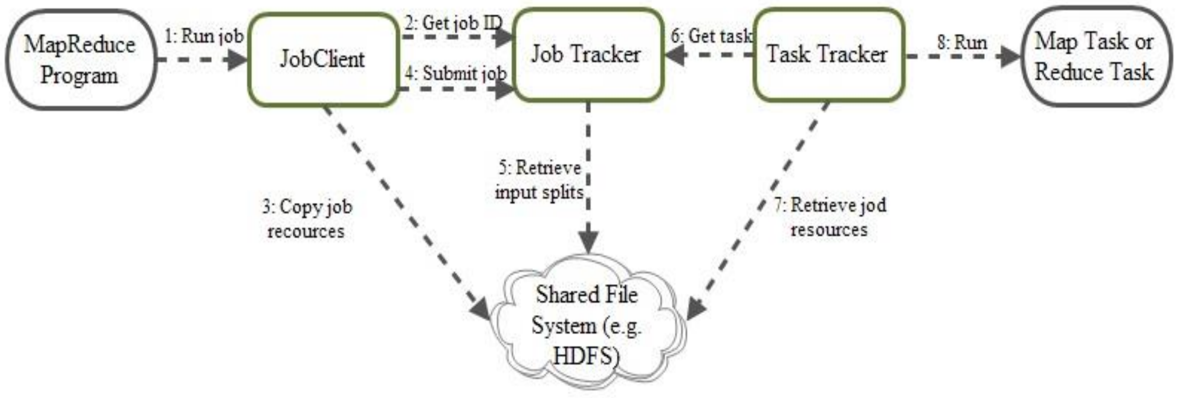


Figure 1.5: Execution flow in MapReduce [15]

The process of running a mapreduce job consist of 8 major steps.

- The first step is the mapreduce that has been written tells the job client to run a mapreduce job.
- This sends a message to the Jobtracker which produces a unique ID for the job.
- JobClient copies job resources, such as jar file containing the java code that has been written to implement map and reduce tasks, to the shared file system, usually HDFS.
- As soon as the resources are in HDFS, the JobClient tells the JobTracker to start the job.
- The JobTracker does its own initialization for the job. It calculates how to split the data so that it can send each split to a different mapper process to maximize throughput. It retrieves these input splits from the distributed file system.
- The TaskTrackers are continuously sending heartbeat messages to the JobTracker. Now that the Jobtracker has work for them, it will return the map task or reduce task as response to the heart beat.
- The TaskTracker need to obtain the code to execute, so they get it from the shared file system.

1.5.3 Inputs and Outputs to MapReduce

The MapReduce framework works only on <key, value> pairs. That is, the framework accepts the input to the MapReduce work 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. Now the work of the reducer is to reduce the input data. Reducer can do that according to various factors. Like it can combine all the values which had the same key. And then it keys the list of values that is list(v2).

1.6 Hadoop

1.6.1 Hadoop Overview

Hadoop[16] is the Apache Software Foundation open source and Java-based implementation of the Map/Reduce framework. Hadoop was created by Doug Cutting. Hadoop originated from Apache Nutch3 which is an open source web search engine was a part of the Lucene project. Nutch was an ambitious project started in 2002, and it soon ran into problems with the creators realizing that the architecture they had developed would not scale to the billions of web-pages on the Internet. But in 2003, a paper was published that described Google's distributed file system - the Google File System (GFS)[17]. Hadoop was born in February 2006, when they decided to move NDFS and Nutch under a separate subproject under Lucene. In January 2008, Hadoop made its own top level project under Apache and HDFS or Hadoop Distributed File System was name kept instead of NDFS. Hadoop provides various tools which help in processing of vast amounts of data using the Map/Reduce framework and, additionally, implements the Hadoop Distributed File System (HDFS).

1.6.2 Hadoop Distributed File System or HDFS

HDFS is a filesystem which is designed for storing very giant files with streaming data access patterns. HDFS runs on clusters on commodity hardware. HDFS was designed keeping in mind the ideas behind Map/Reduce and Hadoop. This implies that it is capable of handling datasets of much bigger size than conventional file systems (even petabytes). These datasets are divided into blocks and stored across a cluster of machines which run the Map/Reduce or Hadoop jobs. This helps the Hadoop framework to partition the work in such a way that data access is local as much as possible.

A very important feature of the HDFS is its "streaming access". Once the data is generated and loaded on to the HDFS, it assumes that each analysis will have a large proportion of the dataset. So the time taken to read the whole of dataset is more important than the latency occurred in reading the first record. This has its advantages and disadvantages. On one hand, it can read bigger chunks of contiguous data locations very fast, but on the other hand, random seek turns out to be a so slow that it is highly advisable to avoid it. Hence, applications for which low-latency access to data is critical will not perform well with HDFS.

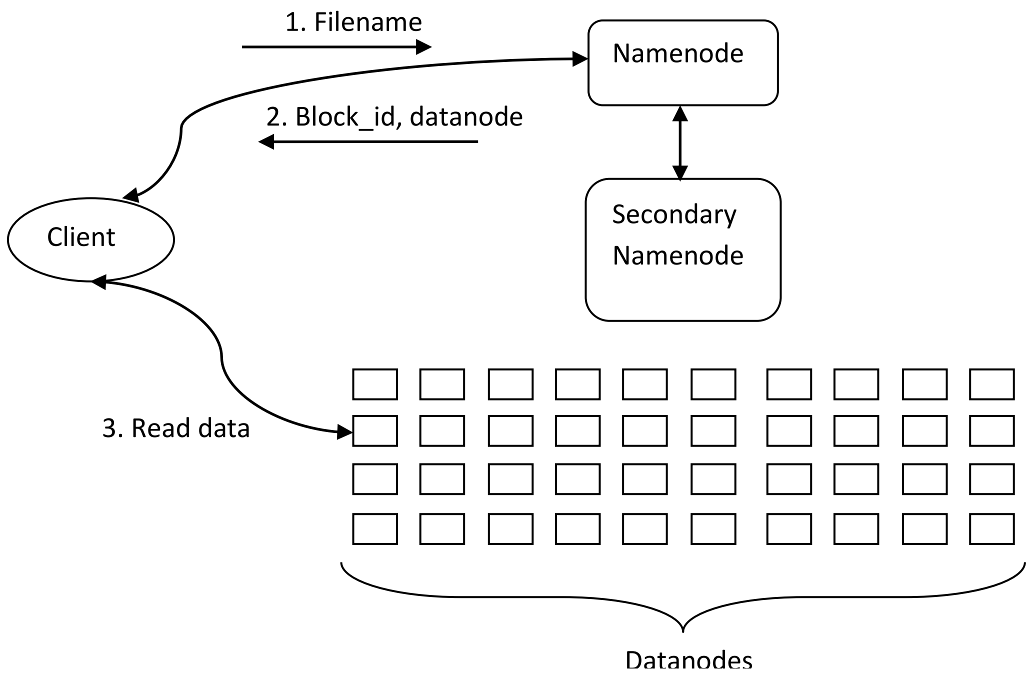


Figure 1.6: Architecture of HDFS [16]

1.6.3 Hadoop Cluster

A Hadoop cluster consists of the following main components, all of which are implemented as JVM daemons.

- **JobTracker**

  Master node controlling the distribution of a Hadoop (MapReduce) Job across free nodes on the cluster. It is responsible for scheduling the jobs on the various TaskTracker nodes. In case of a node-failure, the JobTracker starts the work scheduled on the failed node on another free node. The simplicity of Map/Reduce ensures that such restarts are easily achievable.

- **NameNode**

  Node controlling the HDFS. It is responsible for serving any component that needs access to files on the HDFS. And it is the NameNode only that is responsible for ensuring that HDFS is fault tolerant. In order to achieve the fault tolerance, the replication of files is made over 3 different nodes in which two nodes are from same rack and one node from different rack.

- **TaskTracker**

  Node actually running the Hadoop Job. It requests work from the JobTracker and reports back on updates to the work allocated to it. The TaskTracker daemon does not run the job on its own, but forks a separate daemon for each task instance. This ensure that if the user code is malicious it does not bring down the TaskTracker

- **DataNode**

  This node is part of the HDFS and holds the files that are put on the HDFS. Usually these nodes also work as TaskTrackers. The JobTracker tries to allocate work to nodes such files accesses are local, as much as possible[18].

1.6.4 Data Replication

HDFS is designed to store very giant files across machines in large clusters. Files are stored in a sequence of blocks. These blocks of a file are replicated to provide fault tolerance. The block size and number of replications can be configured per file. Based on the application it can be decided that how many replications would be there of a file. The factor of replication can be specified at the time of file creation time and can be changed later[19].

All decisions regarding replication of blocks is made by the NameNode. It keeps on receiving the heartbeat and blockreport messages periodically from every Datanode in the cluster. As long as it receives the heartbeat messages, it means that DataNode is up and working properly. While the blockreport tells about all the blocks present on the Datanode.

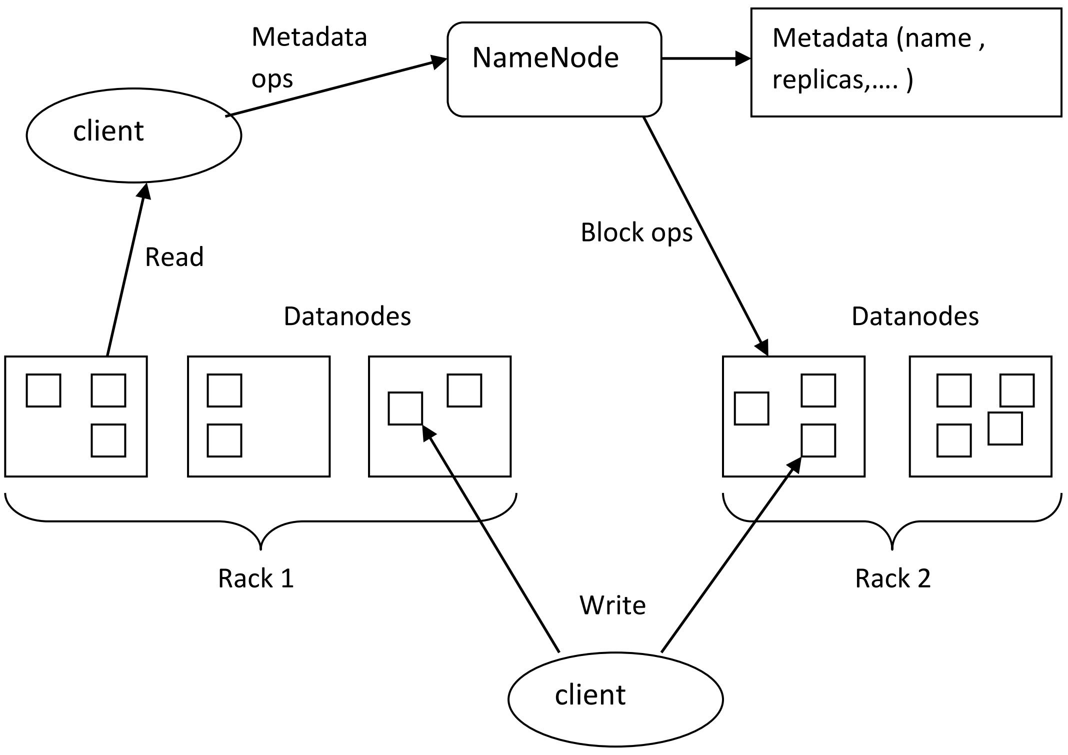


Figure 1.7: Replications in HDFS[16]

Chapter 2
Literature Review

2.1 Geo spatial query in event detection

The rise of Social Media platforms in recent years brought up huge information streams which require new approaches to analyze the respective data. At the time of writing, on social networking sites, more than 500 million posts are issued every day. A large part of these information streams comes from many private users who describe everything on the social media like how they are feeling or what is happening around them or what they are doing currently. It is understood that how to leverage the potential of these real-time information streams.

Many people share the information like what they are interested in. Some are interested in house fires, on-going baseball games, some are interested in bomb threats, parties, Broadway premiers, gatherings, traffic jams, conferences and demonstrations in the area one monitor. Furthermore, independent from the event type, some people like to pinpoint it on a map, so that others could know about it and the information becomes more actionable. So, if there is an earthquake in the area one monitor, one want to know where it caused what kind of casualties or damages[20]. One believes that such a system can be useful in very different scenarios. In particular, one see the following customer groups and use cases:

- Police forces, governmental organizations and fire departments can increase their situational awareness picture about the area they are responsible for.
- Journalists and news agencies to instantly be informed about breaking events. Private customers that have an interest in what is going on in their area.
- Here, the particular nature of Twitter[21] and its adoption by a younger, “trendy” crowd suggests applications along the lines of, e.g., a real-time New York City party finder, to name just one possibility.
- So based on all those streams event detection will take place.

2.2 Optimization of MapReduce Jobs to Improve the Large-Scale Data Analysis Process

Data-intensive applications works on large amount of data using any parallel processing paradigm. MapReduce is a programming model made for data-intensive applications which contain massive data sets as well as execution framework for processing of large-scale data on clusters. While there are many strong points about MapReduce like its easy programming structure, high scalability, fault tolerance, but its configuration parameters need to be finely and efficiently tuned for the specific deployment. MapReduce performance is also directly dependent on the various configuration parameters of Hadoop in various conditions. So in order to achieve maximum performance all these configuration parameters should be tuned very carefully[22].

Therefore tuning requires mainly 3 steps

- Data redistribution technique should be extended so as to find the nodes with high performance.
- Number of map/reduce slots should be utilized so as to make the execution time more efficient.
- Making a new routing schedule shuffle phase so as to define the scheduler task while memory management level has reduced.

Example: The extended methods added to MapReduce have become widespread in recent years. A main objective of such a system is to improve the performance of complex data analytical tasks and confirm the potential of their approach[23] which is one of the key technologies for the extension and improvement of the MapReduce runtime framework to efficiently process complex data analysis tasks on large clusters. Suppose a data analysis task requires the joining of multiple data sets to compute certain aggregates. Various technologies that are being used are

- **Filtering–join–aggregation**

  Suppose that the approach requires a filtering–join–aggregation task in two successive MapReduce jobs. Filtering logic is applied by the first job to all the data sets in parallel, joins the qualified tuples, and sends the join results to the reducers for partial aggregation to improve the performance of complex analysis queries.

- **Map-reduce-merge**

  Map–reduce–merge included a merge phase on top of MapReduce for increasing the efficiency of the merged data, which has already been partitioned and has been sorted (or hashed) by the map and reduce modules.

- **MapReduce-indexing**

  MapReduce-indexing strategies provided a detailed analysis of four MapReduce indexing strategies of varying complexity and were examined for the deployment of large-scale indexing.

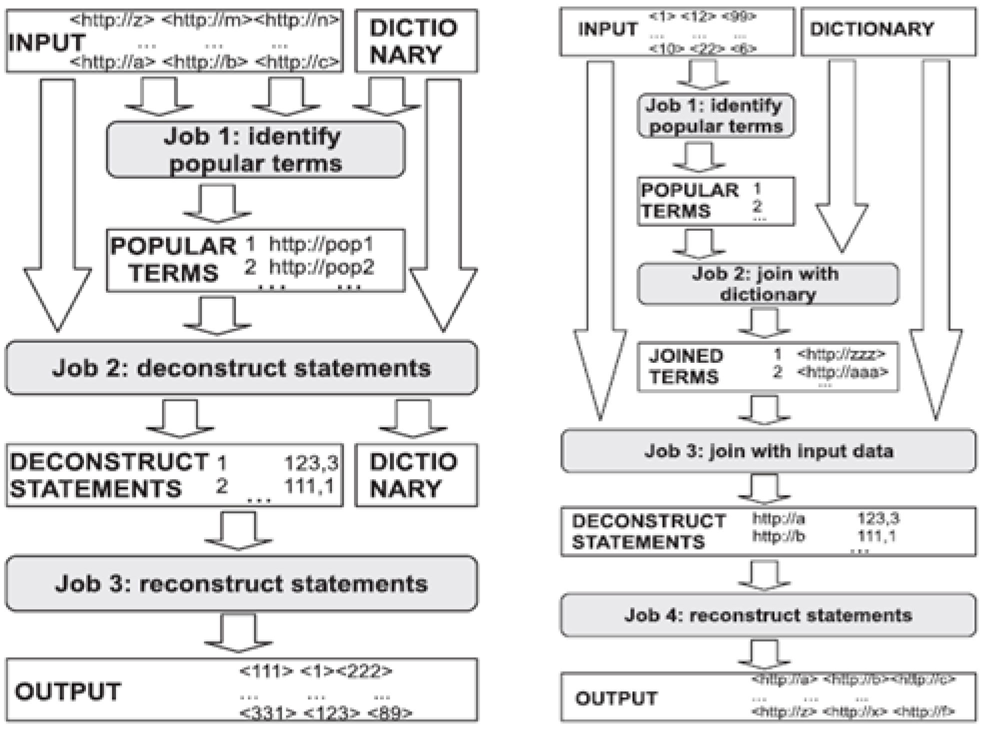
- **Pipelined-MapReduce**

  The pipelined-MapReduce allows data transfer by using a pipeline between the operations and it expands the batched MapReduce programming model, and reduces the completion time of tasks and improves the utilization rate.

One of the most important requirements for effective performance tuning is to discover those important parameters that are related to tuning a job for all features. Therefore, the following question naturally arises: there are good approaches for optimizing all the problems of MapReduce. All of these studies demonstrate the usefulness of the performance-tuning job of MapReduce. These are the main parameters which need to be tuned in any MapReduce job so as to improve the efficiency, so these parameters need to look after first of all.

2.3 Scalable data compression with MapReduce

The Semantic Web[24] consists of billions of statements. These statements are released due to usage of RDF (Resource Description Framework) [25] data model. In order to handle so much big amount of data, a compression technique is required to be applied by the RDF. But that size of data is so big that even applying the compression to such data is difficult. Therefore in order to do efficient compression and decompression of such big data MapReduce algorithms are used. It makes the use of dictionary encoding technique. Dictionary encoding technique maintains the structure of the data. The SemanticWeb is an extension of the current WorldWideWeb, where the semantics of information can be interpreted by machines. Information is represented as a set of resource description framework (RDF) statements, where each statement is made of three different terms: a subject, a predicate, and an object. An example statement is <http://www.vu.nl> <rdf:type> <dbpedia:University>. This example states that the concept identified by the uniform resource identifier (URI). Flow diagram for compression and decompression of RDF.


INPUT
<http://z> <http://m><http://n>
<http://a> <http://b> <http://c>
DICTIONARY
Job 1: identify popular terms
POPULAR TERMS 1 http://pop1 2 http://pop2
Job 2: deconstruct statements
DECONSTRUCT STATEMENTS 1 123,3 2 111,1
DICTIONARY
Job 3: reconstruct statements
OUTPUT
<111> <1><222>
<331> <123> <89>
INPUT <1> <12> <99> <10> <22> <6>
DICTIONARY
Job 1: identify popular terms
POPULAR TERMS 1 2
Job 2: join with dictionary
JOINED TERMS 1 <http://zzz> 2 <http://aaa>
Job 3: join with input data
DECONSTRUCT STATEMENTS http://a 123,3 http://b 111,1
Job 4: reconstruct statements
OUTPUT
<http://a> <http://b><http://c>
<http://z> <http://x> <http://f>

(a) Compression algorithm (b) Decompression algorithm

Figure 2.1: Compression and decompression of RDF[26]

2.4 Finding nearest locations using zip code

Nowadays almost every Restaurant and other businesses have a functionality of "Find Locations" on their websites which gives nearest locations for a given address or Zip. Now the methods which is used behind this query is that it matches the zip code of all places one by one and the code of which places fall under same zip code , it give them as their closest locations. This is done by matching the zip code with others in the Database. So those places whose zip will match with the zip of the hotel or restaurant will be displayed. It is obvious that places which will fall under same zip would belong to the same region and would be near to each other.

Example Like one has to find the restaurants near sector-23 of Chandigarh and zip code of this sector is 160023. Now what this query will do is that it will match the zip code of all the restaurants in the database to that of sector-23 that is 160023, and the restaurants which will have same zip code to 160023 will be given as the nearest restaurants. These figures show the current location of a person in encircled dot and all the other location of various places as simple dots.

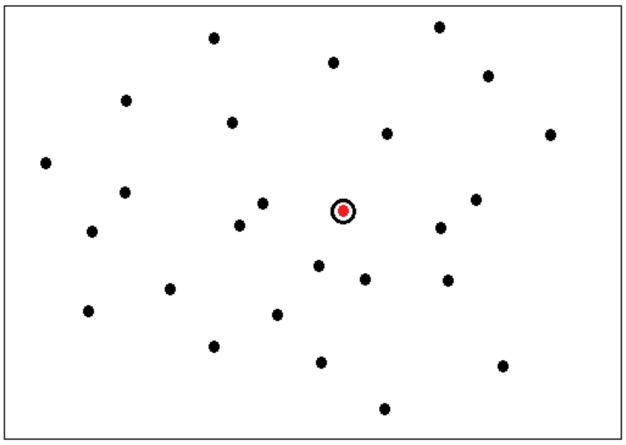


Figure 2.2: Locations across a certain region in zip code method

Figure 2.2 shows encircled dots (our current location) surrounded by various other dots(different places) which can be compared to a real life scenario. These circles depict various restaurants, theatres, libraries, temples etc. Now we want to find the places near to our locations, so what this method will do that it will show all the places that match the zip code of the current location.

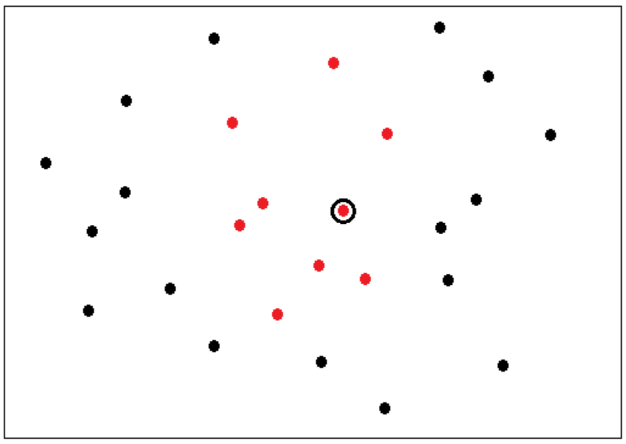


Figure 2.3: Nearest locations found through zip code method

So figure 2.3 shows the places which had the same zip code as our current location and are depicted as red in it.

2.5 Finding the nearest locations using polygons

In some of the geospatial frameworks, a zip code is represented by a "polygon". A polygon is an area defined on the map and it has clear boundaries. One can also define new polygons based on the original polygons in these frameworks. So a new polygon can be created dynamically that will be the zip code or it can be extended by any much at any point. So now the shape created after defining the polygon will be taken into account. It is easy with an address because just a polygon has to be created, which is a circle around one point. Then queries can be done which will be based on all the new points that have been defined inside the new polygon.

This is the method which is adopted by most of the sites. They just display all the points or locations that fall within the specified boundary. And they themselves define this polygon which can be of 5 km or 10 km. They do not actually calculate the distance, but just giving the locations of places that fall within that polygon. So we can define any polygon of our choice n see our desired location in that much area.

Example: Like one has to find all the nearby temples to Thapar University in Patiala. Now it is known that in this method a zip code is defined as a "polygon" and the polygon defines an area on the map having clear boundaries. So these methods will first induce the boundary of the zip code of the given location and then will give all the places falling within that boundary.

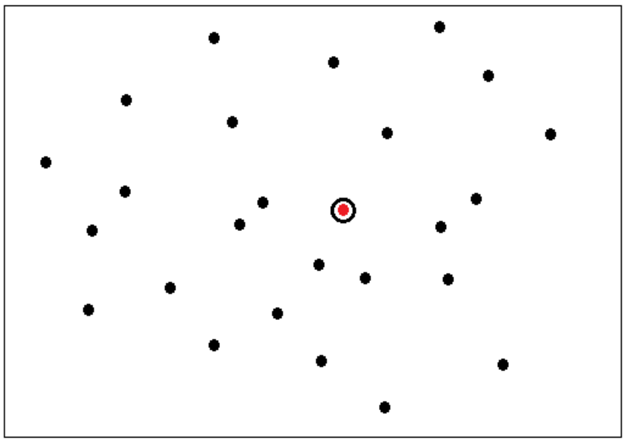


Figure 2.4: Locations across a region in polygon method

Figure 2.4 shows encircled dots (our current location) surrounded by various other dots (different places) which can be compared to a real life scenario. These circles depict various restaurants, theatres, libraries, temples etc.

Now one has to find the location of various nearby places. So all places falling within that polygon will be shown as output.

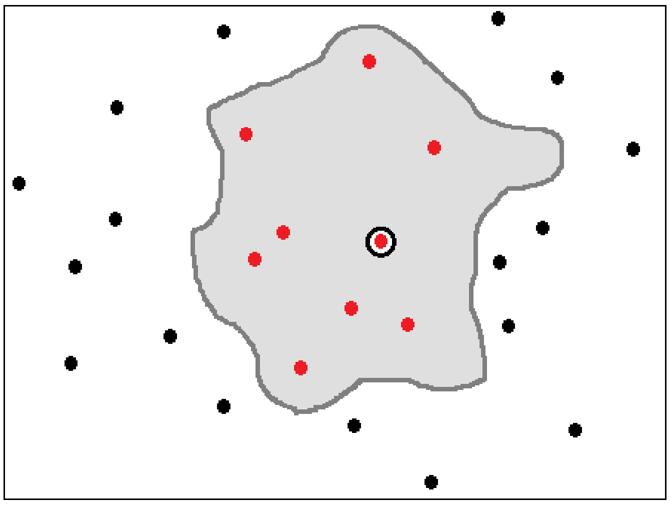


Figure 2.5 : Nearest locations found by polygon method

Figure 2.5 shows default polygon depicting the zip code of an area. All the places within this polygon are shown as red dots and our required places.

In this method one can also dynamically define his own polygon, like suppose that the radius of the default polygon was 10 km. Now he can define the polygon which will have radius of 5km.

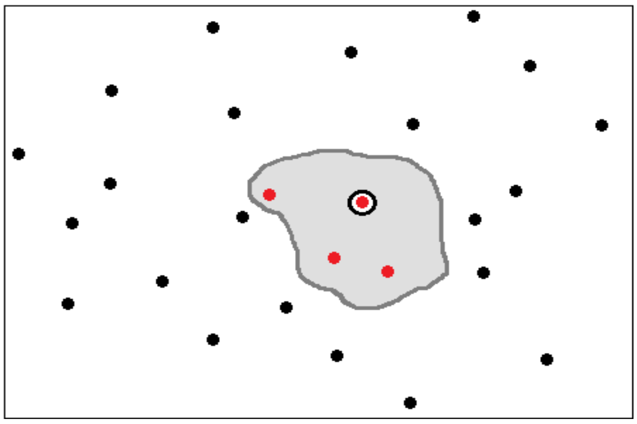


Figure 2.6: Locations inside a small polygon

Figure 2.6 shows the dynamically defined polygon with a radius of 5km and red dots within it define all the places that fell within this region.

2.6 Finding nearest location using voronoi diagrams

A Voronoi diagram divides space into a number of regions. To make the voronoi diagram ,first the numbers of points are specified, which are called sites or seeds. And then based on those seeds a region is made around every site. Those regions depict that all the points in that region will be the nearest to that particular site than any other site. The regions defined are called Voronoi cells. Therefore there will be a clear picture of the subdivided region such that every region has its own closest seed or site. The Voronoi diagram having a set of n point sites in the plane can be computed in O(nlogn) time[27].

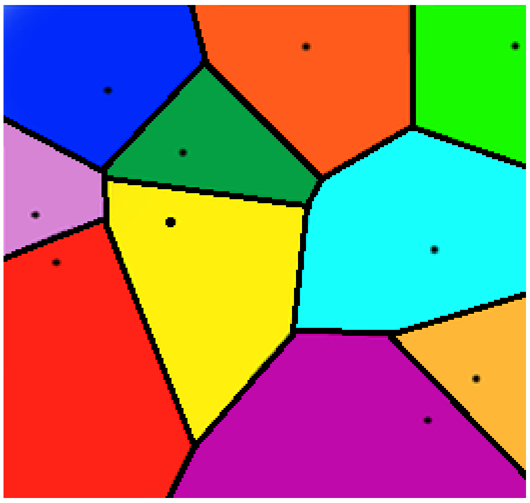


Figure 2.7: Voronoi diagram

Voronoi diagram construct various regions based on the input points. Let those points depict any place like hotels in a city. Now voronoi diagram will convert that into small regions such that for every region it has its own nearest hotel. Or it can be said that every hotel will be assigned its own region, in that region all the points will have this hotel as their nearest hotel.

- **How to find nearest neighbor**

Here is a pictorial view of all the points which are depicting any point of interest, let it be hotels. Now we are given our current location and we need to know the nearest hotel. Now we will check that our location is falling under which region.
Hotel in that region will be the nearest hotel.

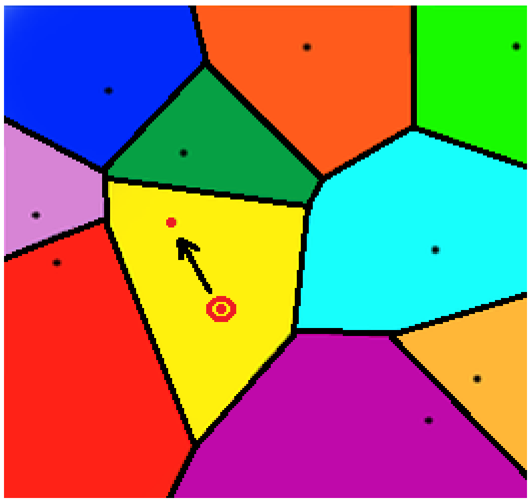


Figure 2.8: Nearest location through voronoi method

2.7 Finding nearest neighbor through k-d trees

The k-d tree is a binary tree in which every vertex is a k-dimensional point. Every non-leaf vertex can be taken as spitted hyper-plane that makes two partitions of space that known as half-spaces. The points in left side of this hyper-plane are depicted as the left subtree of that vertex and points in right side of the hyper-plane are depicted by the right subtree[28]. The direction of hyper-plane is selected in the following way:

- Every vertex in the tree is related with one of the k-dimensions, axis of dimension is perpendicular with the hyper-plane.
- For example, if for a specific division the “x” axis is selected, all points in the subtree with a smaller “x” value than the vertex will appear in the left side subtree and all larger “x” value points will be in the right side subtree.
- In that type of case, the hyper-plane can be the set by the x-value of the point, and normal of it can be the unit x-axis.

2.7.1 Construction of k-d trees

To add the first vertex into the k-d tree, in array 0 the median element is chosen and stored it in the tree vertex. An array 0 is divided into two sub-arrays using median element. Above the median element stores in one sub-array and the other sub-array lies below median element. The x-coordinate of median element defines an x-aligned is dividing a plane that may be used to divide each of the other k-1 arrays into two sub-arrays.

The following steps divide an array into two sub-arrays:

- Arrange each element of the array in first to last form.
- Test against the dividing plane, the array element refers to the x-coordinate of the point.
- Now assign that element to one of two sub-arrays, it all depends on the point which side of the dividing plane it reside.
- Ignore that array element which refers the same point that the median element of array 0 refers, because this point represents the dividing plane.

This step divides the arrays into two sets of sub-arrays while retaining the original sorted order within each sub-array. The procedure does in recursive manner to add vertices into the two sub-trees at the next level of the tree using these sub-arrays.

These procedures will simplify design of k-d trees:

- Arrays should be divide into sub-arrays that defines “greater than or equal to” and “less than” partitioning. After selecting the median element of array 0 this conversion needs that the element of array 0 that lies immediately below the median element be examined to ensure that this adjacent element refers a x-coordinate of point whose is less than and not equal to the x-coordinate of the dividing plane.
- If this adjacent element refers a point whose x-coordinate is equal to the x-coordinate of the dividing plane, continue finding from the starting of array 0 until the first instance of an array element is got that refers a point whose x-coordinate is less than and not equal to the x-coordinate of the dividing plane. When this array element is got, the element that lies immediately above this element is the

correct choice for the median element. Apply this method of selecting the median element at each stage of recursion.

- This procedure for producing sub-arrays assures that the two sub-arrays contain one less array element than the array from which these sub-arrays were generated. This classification permits reuse of the k arrays at each stage of recursion as follows:

1. Copy array 0 into a temporary array
2. Create the sub-arrays that are produced from array 1 in array 0
3. Create the sub-arrays that are produced from array 2 in array 1
4. Continue this pattern, and create the sub-arrays that are produced from array k-1 in array k-2
5. Finally copy the temporary array into array k-1. This method permutes the sub-arrays so that at successive stages of the k-d tree, the median element is selected from x, y, z w, sorted arrays.

Let the points on our x-y axis be

point_list = [(2, 3), (5, 4), (4,7), (9,6), (8, 1), (7, 2)]

Now form a k-d tree of it using the above guidelines. So now the procedure can be shown as.

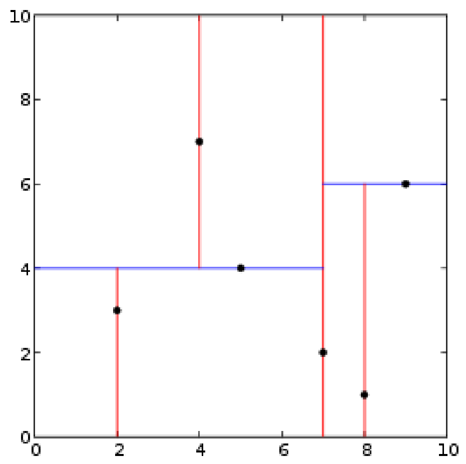


Figure 2.9: Procedure to make k-d tree

And the tree which will be formed after following this procedure is shown as:

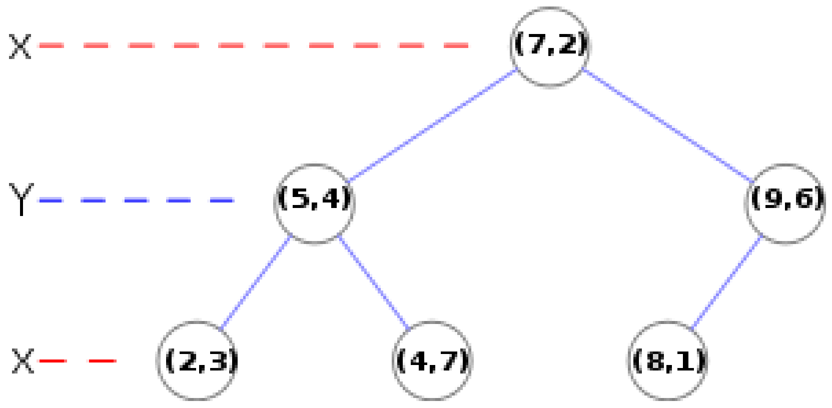


Figure 2.10: k-d tree

2.7.2 Applying nearest neighbor search on k-d tree

Now that we have formed our k-d tree, we can now easily find our nearest point of interest using this tree. This searching has become much efficient now and has the time complexity of log(n). The goal of nearest neighbor search (NN) algorithm is to search the point in the tree that is close to a user input point. This search can be performed well by using the tree properties to fast remove large portions of the search space.

Search for a closest (nearest) neighbor in a k-d tree is done in following ways:

1. Starting with the root vertex, the algorithm perform on the tree recursively, in the same fashion that it would if the search point were being added (i.e. it goes left or right depending on whether the point is less than or greater than the current vertex in the split dimension).
2. Once the algorithm reaches a leaf vertex, vertex point saves as the "current best"
3. The algorithm unwinds the recursion of the tree, performing the following steps at each vertex:
    1. If the current vertex is closer than the current best, then it set as the current best.
    2. The algorithm checks whether there could be any points on the other side of the dividing plane that are closer to the search point than the

current best. This is performed by intersecting the dividing hyper-plane with a hyper-sphere around the search point that has a radius equal to the current nearest distance. Since the hyper-planes are all axis-aligned this is implemented as a simple comparison to see whether the difference between the dividing coordinate of the search point and current vertex is less than the distance (overall coordinates) from the search point to the current best.

    1. If the hyper-sphere crosses over the plane, there could be nearer points on the other side of the plane, so the algorithm would step down the other branch of the tree from the current vertex looking for closer points, following the same recursive steps as the entire search.
    2. If the hyper-sphere doesn't intersect the diving plane, then the algorithm continues step up the tree, and the whole branch on the other side of that vertex is eliminated.

4. When the algorithm finishes this process for the root vertex, then the search is complete.

Chapter 3
Problem Statement

3.1 Proposed objective

This algorithm will 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.

So as this algorithm will find the nearest location there will 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. In MapReduce execution takes place in parallel, so also it will take less in executing the large database. Mapreduce is mainly useful in those cases where large amount of data is involved, so it well suits our work.

Many methods are already there to find nearest location, but somewhere all of them have few or more flaws which are not well suited for our application. These various drawbacks are defined as follows with respect to the existing methods.

3.2 Gap analysis

As various methods or algorithms to find the nearest neighbours have been seen. They all are good in particular situations but may fail or might be in efficient in other scenarios.

So now we will have a look at the gaps between those methods one by one as compared to our proposed method.

- **Gaps in Zip code**

It was found that in zip code method if anyone want to find the nearest location with respect to his current location then it will show all the locations with zip code same as the our current location.

But this method may fail in a various scenarios. First of all it gives range of location not just one nearest location and it doesn’t give the accurate answer. It just considers the fact that locations in the same city or same zip code are mostly nearest to each other.

Let us understand this fact of failure by taking an example.

Let us say that user is standing at some location in the city which is at the corner of the city. And we want to find the nearest temples.

Now this method will give all the temples those falls in the same city as they all have same zip code. But it is not the correct solution according to this scenario because we are standing at the corner of the city, so temple of other city might have been closer but it didn’t consider that fact.

It can be understood through figures as:

Figure 3.1 shows the current location which is depicted by encircled dot. And all other dots show the location of all other temples in the city and around that city. The shaded portion depicts the area of the city. Current location is just at the edge of city.

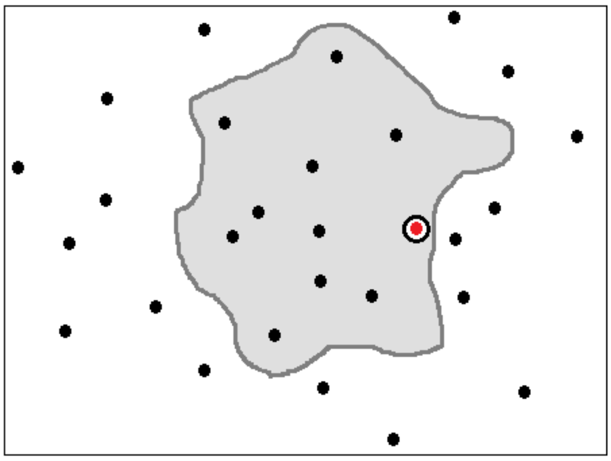


Figure 3.1: Gaps in zip code method

Figure 3.2 shows the location which was given as output of the query to find nearest temples. So it showed all the temples of the city instead of showing that temple pointed by arrow is more nearer.

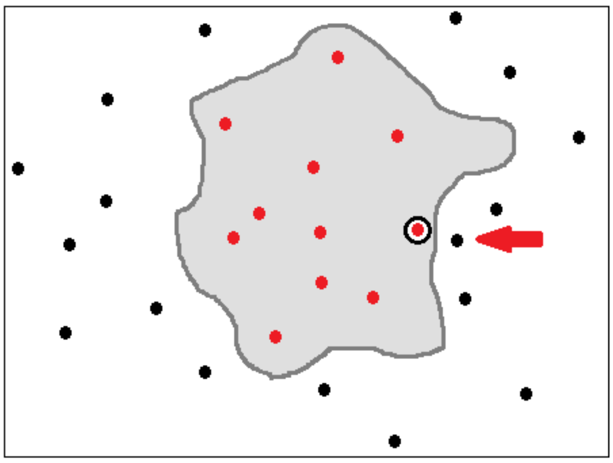


Figure 3.2: Gap in zip code method

- **Gaps in Polygon method**

Polygon method considers the zip area of a city as a polygon, so it gives the output based on that area. So if one puts a query to find any location then it will reply based on the polygon in which our current location is.

It doesn't take any distance into consideration too. By default polygon is defined as the area of the city but one can also draw her own polygon of desired area. In this way one can be more precise if she keeps on decreasing the area of polygon. But there is no surety that she will get the nearest location. One may think that she can get the nearest location by continuously decreasing the area of polygon until she gets only one location and which is her nearest location.

But this is not a practical approach to find the nearest location and doesn't also work always. It is just practical to find a range of locations. One might have to draw large number of polygons with slighter difference between them in order to get desired result. And also this approach also gives the range of locations but not just one nearest location.

Other than this one also needs a pre well defined map on which this method will be applied, just a database of locations wouldn't solve the purpose.

- **Gaps in K-d trees**

K-d trees works on the points given in the 2D plane. First those points are converted into K-d tree and then that K-d tree is used to find the nearest neighbor of any point. K-d tree ensures a tree depth of log(n) to fasten the search.

But problem with k-d tree is that it cannot be used for search of our nearest location, because k-d tree works on points like points on 2D plane from which it calculates distance between them and so on.

These points can be think of as (longitude, latitude) of a location but these (longitude, latitude) points cannot be represented in a 2D plane because the distance between two particular longitudes doesn’t remain same throughout.

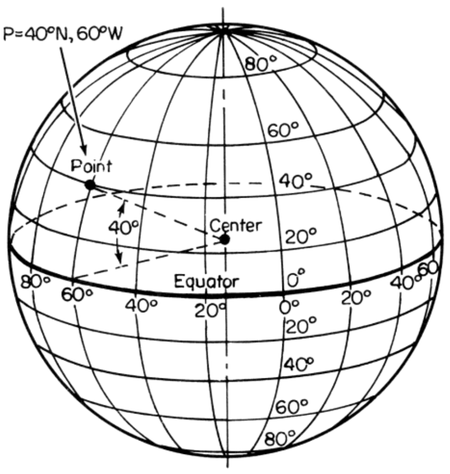


Figure 3.3: Longitude and latitudes of earth

So we cannot apply this algorithm there. Moreover if we think to save the actual distances between every two places in the database, then it is not a practical approach because our database would be too big.

- **Gaps in Voronoi diagrams**

A Voronoi diagram is a way of dividing space into a number of regions. A set of points (called seeds, sites, or generators) is specified beforehand and for each seed there will be a corresponding region consisting of all points closer to that seed than to any other. The regions are called Voronoi cells. Those points can be any location in the city like theatres, pubs, schools, hotels etc. When one want to find the nearest location from any current location then it is seen that current location is falling under which region. Now the point of that region around which the region was drawn is chosen as the nearest location.

Now the drawback in this method is that first of all it is not practical to draw a very big voronoi diagram like which is covering whole of the nation or large part of it. It will be of no use in that case. It is only useful for smaller areas like for one city, or some more cities. But it cannot be drawn for unbounded data.

Secondly there are so many categories of locations like schools, libraries, stadium, hotels, temples, lakes and so on. One might put a query to find anyone of that. In the case of voronoi diagrams it will be needed to draw a different voronoi diagrams for every different category because all the locations cannot be shown in one voronoi diagram. So if a voronoi diagram is drawn for any city, there would be so many voronoi diagrams for it, one for each category, and when one wants to find any nearest location, he will have to find voronoi diagram of that particular category. It would be so heavy on the database.

So it can be seen that none of the methods have been implemented with MapReduce yet and moreover they are not solving the problem in any accurate way. So the proposed solution given will be much efficient with respect to time and accuracy both.

Chapter 4
Implementation

4.1 Overview

The solution is provided to this problem using MapReduce. MapReduce executes in parallel, so our large data will be processed parallelly altogether. MapReduce needs two functions to be made namely Map and Reduce.

So all the work is carried out in these two functions only. Map function will take the input in the form of <key, value> pair and provides a new <key ,value> pair as output. This output then goes to reducer which then processes this data and gives the output.

- Here <key, value> pair will be <long :lat, type:place>
- Long stands for longitude
- Lat stands for latitude
- Place stands for name of the location
- Type stands for category of the place

Now every location has its own unique (longitude, latitude) pair, so we will represent the location by its (longitude, latitude) pair in our implementation.

Name of the places will be required at the time of displaying the names in the output.
Type tells that what is the category of the that place like there can be so many categories school, library, hospital, pubs, colleges, temples, museums etc. So the category in which one is interested is put and the all the places belonging to that category are shown as the output in their increasing order of distance from the current location. Map and reduce functions are defined as.

4.2 Algorithms

Algorithm 4.1 Mapper

```
Procedure Map (input_lon, input_lat ,place_type)
1.  geocode ← encode(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 ← encode(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
```

Here first of all, the map function will receive the <key, value> pairs in the form of <longitude:latitude, type:place>

Longitude and latitude are given in the decimal format. Now these (longitude, latitude) combinations are converted into the binary format using the Geohash function. In geohashing by interleaving latitude and longitude information in a bitwise fashion, a composite value is generated that provides a high resolution geographic point, and is well suited for storage or transmission as a character string. Geohash also has the property that as the number of digits decreases (from the right), accuracy degrades. This property can be used to do unbounding box searches, as points near to one another will share similar Geohash prefixes.

However, because a given point may appear at the edge of a given Geohash unbounding box, it is necessary to generate a list of Geohash values in order to perform a true proximity search around a point. Because the Geohash algorithm uses a base-32 numbering system, it is possible to derive the Geohash values surrounding any other given Geohash value using a simple lookup table. That is why geohashing helps to make the search for nearest location easy.

There two functions in geohashing namely encoding and decoding. Encoding converts the (longitude , latitude) pair into binary numbers which is represented in base-32 format and other one is decoding which converts the base-32 format again into the combination of (longitude, latitude).

Now 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 encode function and save it in variable geocode. Now the geohash location index of the current location has been got. 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 the current location. Various categories of locations have been included which can be the point of interest .Various categories which have been used in the implementation are

- ATM
- Bar
- Bus station
- Café
- Fuel
- Library
- Pub
- Restaurant

Separate file for each category has been saved. So the mapping function takes the input from sys.stdin where stdin stands for standard input. Inputs are saved in the format: Longitude, latitude, place_type ,place_name.

So there are total 4 entries in every line each for longitude , latitude , place type and place name. Now we take these lines one by one. Line.strip() function is used for that. Now those 4 entries in each line are splitted which are separated by ",". After that the length of the array "data" is counted in which all the entries of a line had been saved. Only those lines are considered which have 4 complete entries. Some of the lines might be incomplete and don't contain the full information, so we will skip these type of lines .After that the longitude and latitude of each line will be saved in "lon" and "lat" respectively from array elements data[0] and data[1].

Then a variable "count" initialized to 0 will be taken, then will do prefix matching of each line with our input value. It will be seen that up to what prefix length our values match. If prefix matches to the full length then it means that a very high precision has been got and one is standing almost near to the destination location for which he is searching.

Now at the end of map function prefix matched value is got with every entry in the database. So the map function will out put the values in the form of: Length of Prefix matched, place_type, place_name

Longitude, latitude or geohash code is no longer needed because all the sufficient information needed to display the names of nearest locations is already there. So the information about the places that which places are near and which are far is got based on the matched prefix. So based on that information name and type of the place will be displayed.
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.

Algorithm 4.2 Reducer

```
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 the geohash code because only names will be needed now to display them. Mainly filtering is being done in the reduce function. 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. No separate sorting is required to be done in the map function.

Entries corresponding to every 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 to the type provided by the user then it will be displayed. User can 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 is sorted order because hadoop did it itself. 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 4.3 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 doing nearest location searches. 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 among all those places. Geohash algorithm uses a base-32 numbering system.

4.3 Explanation with Example

There is any place in india whose longitude and latitude is given as <38.897, -77>. Now this <longitude ,latitude> combination is encoded into a single entry called the geohash code. And the geohash code that came is: **dqcjqcp84c6e**

It will be matched with every entry in the datafile which are also present in the geohashed form in the datafile. Let the various geohashed entries in the datafile be.

dqcjqcp72c5d

dqcp5aq62d5c

dqcjeqc90q9r

dqcpeqc91q92

dqcjecc8qq9r

dqcjeqc90q9r

dqcgqcg89c6e

dqcjqcp84c6e

dpdkqcd39d2e

These entries will be compared one by one with “**dqcjqcp84c6e**”. First of all it will be tried that whether all its number matches with any entry, in that case it will be very nearest to that place.

But as it not so, therefore the last bit of this code is truncated and becomes “**dqcjqcp84c6**”. Now it is again compared with the entries to see whether this prefix match with prefix of any other entry, but it is not again so.

Now the code is again truncated and becomes “**dqcjqcp84c**” and again matched to see whether this prefix matches with any prefix of the entries. So this truncation process keeps on going until a match is found.

Like here when it will be truncated up to “**dqcjqcp**”, then it will match with the location “**dqcjqcp72c5d**”.

So it is the nearest location which was being searched. And it is the only most nearest location to the specified location. In some cases there can be more than one nearest locations like when two entries whose prefix get matched with the specified prefix. But in this case we got only one nearest location.

4.4 Flowcharts

4.4.1 Flowchart of Mapper

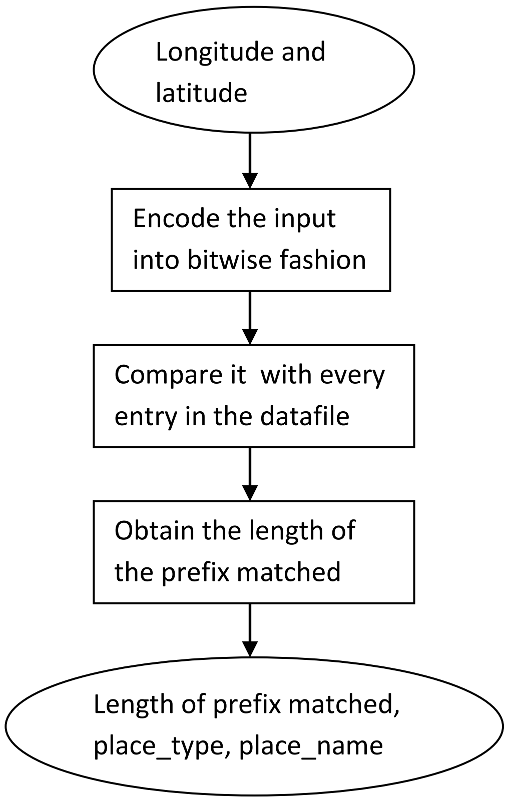

Longitude and latitude
Encode the input into bitwise fashion
Compare it with every entry in the datafile
Obtain the length of the prefix matched
Length of prefix matched, place_type, place_name

Figure 4.1: mapping task

The flow diagram shows the functioning of a mapper in a simple way.

- First of all mapper receives the input in the form of longitude and latitude of the place from the nearest location has to be found.
- The mapper will encode this longitude latitude pair into a geohash code .the geohash code is in bitwise fashion but is represented in base-32 format.
- Now the mapper compares this input geohash code with every other geohash code present in the datafile.
- In the process of comparison to every entry in the datafile, it gets the length of prefix matched to every entry.
- Mapper then passes this length of prefix matched, type of place that is needed to be searched upon and place names to the reducer.

4.4.2 Flowchart of Reducer

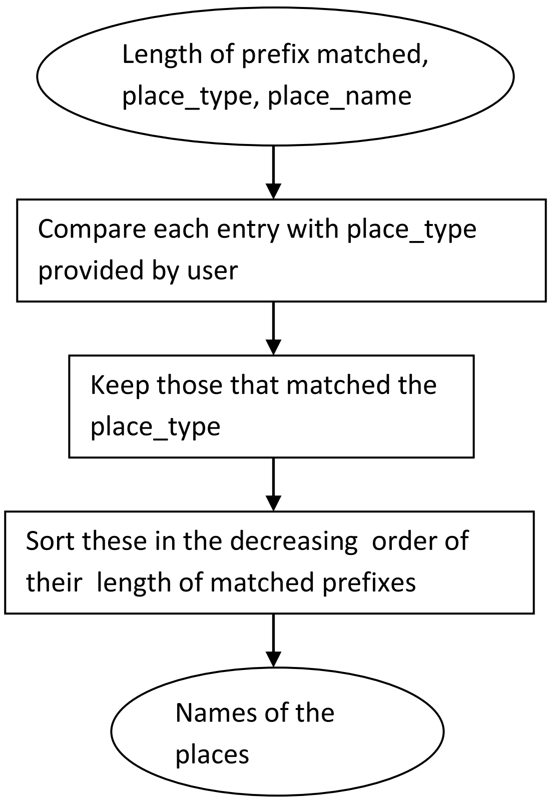

Length of prefix matched, place_type, place_name
Compare each entry with place_type provided by user
Keep those that matched the place_type
Sort these in the decreasing order of their length of matched prefixes
Names of the places

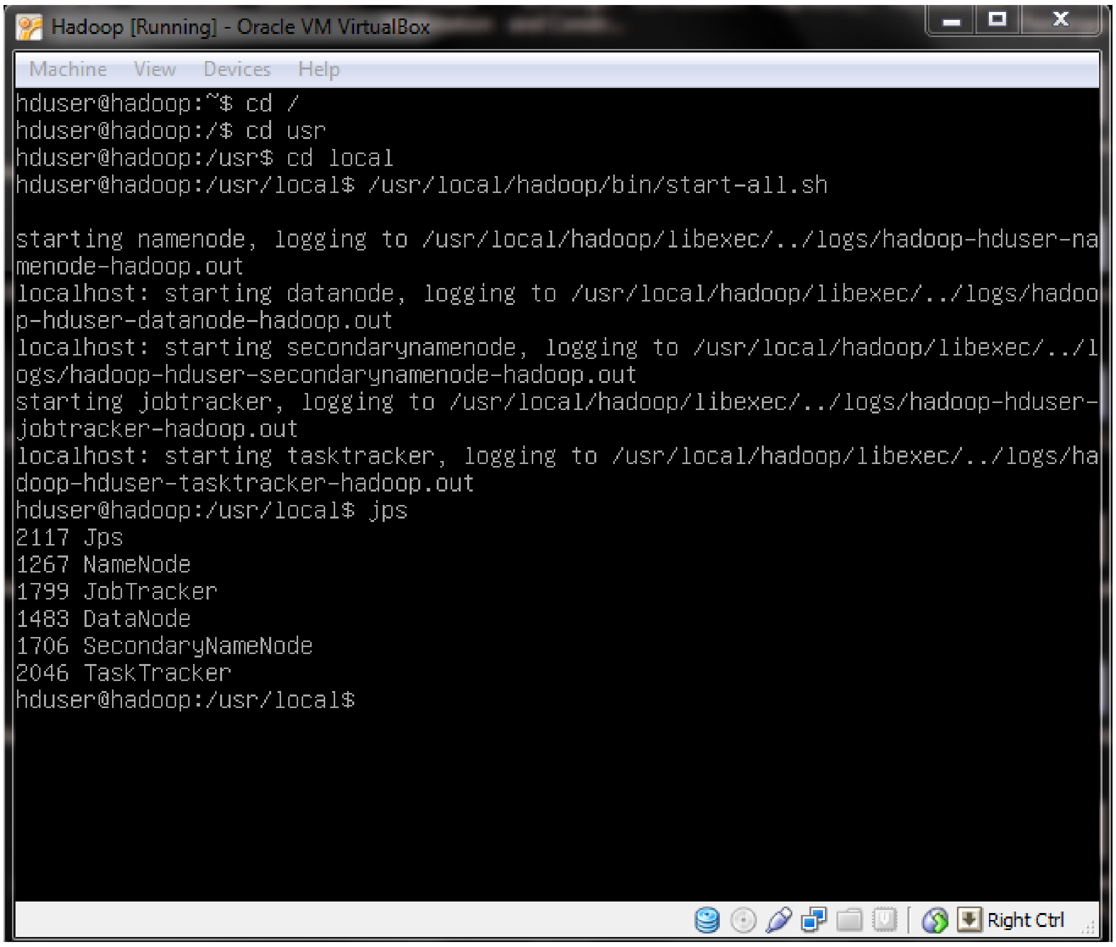
Figure 4.2: Reducing task

Functioning of reducer is defined as

- Length of prefix matched of each entry in the datafile, type of place entered by the user and name of places which were given as output by the mapper are received as input by the reducer.
- Reducer has now to do the filtering. For this it will compare the each incoming entry with the type of the place entered by the user so as to keep only those entries in which user is interested.
- Now that only those entries in which user is interested have remained, mapper will sort them in decreasing order of their length of prefix matched.
- In the end the names of all these places is got as the output. The output is get in the form of a list in which top most entry depicts the nearest place and bottom one depicts the farthest.

Chapter 5

Results and Analysis

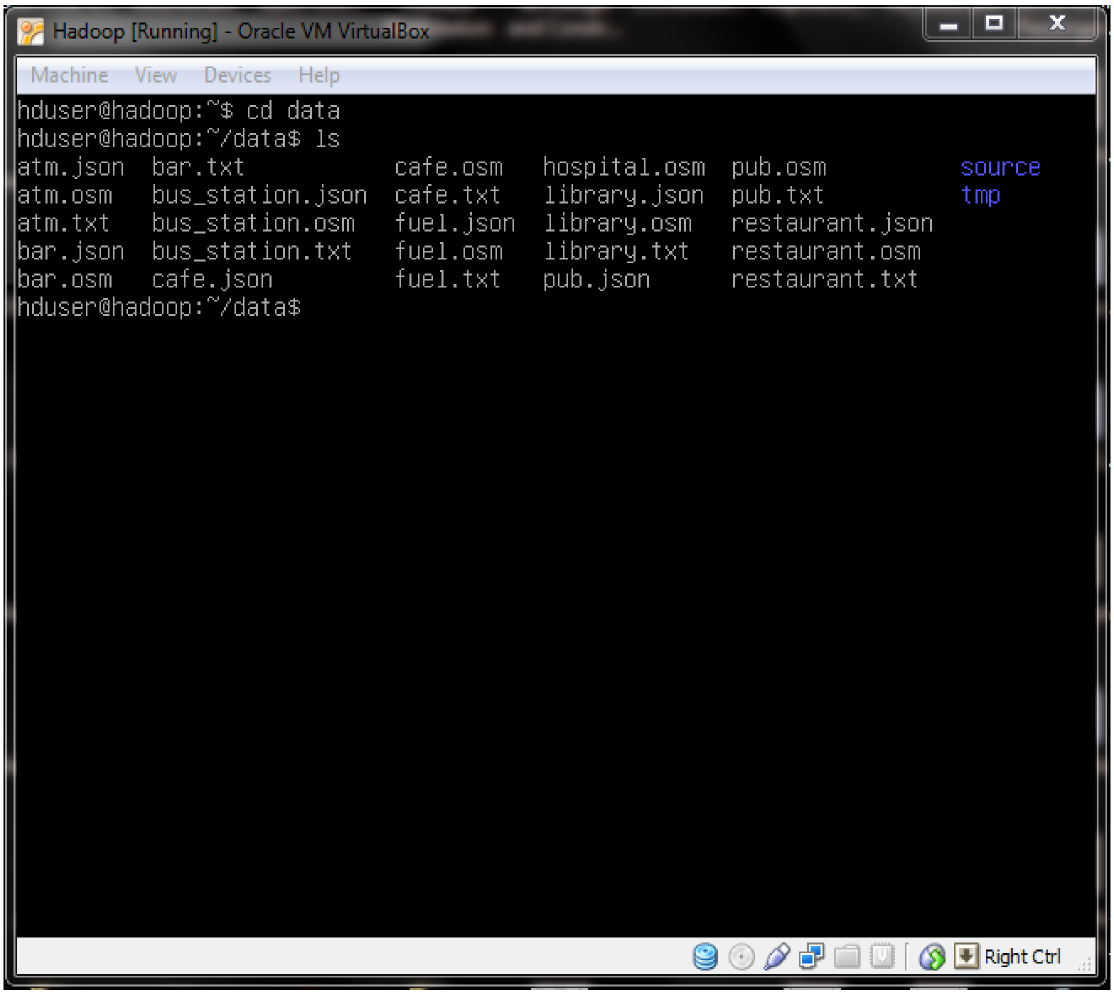

Hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
Right Ctrl

```
hduser@hadoop:~$ cd /
hduser@hadoop:/$ cd usr
hduser@hadoop:/usr$ cd local
hduser@hadoop:/usr/local$ /usr/local/hadoop/bin/start-all.sh

starting namenode, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-na
menode-hadoop.out
localhost: starting datanode, logging to /usr/local/hadoop/libexec/../logs/hadoo
p-hduser-datanode-hadoop.out
localhost: starting secondarynamenode, logging to /usr/local/hadoop/libexec/../l
ogs/hadoop-hduser-secondarynamenode-hadoop.out
starting jobtracker, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-
jobtracker-hadoop.out
localhost: starting tasktracker, logging to /usr/local/hadoop/libexec/../logs/ha
doop-hduser-tasktracker-hadoop.out
hduser@hadoop:/usr/local$ jps
2117 Jps
1267 NameNode
1799 JobTracker
1483 DataNode
1706 SecondaryNameNode
2046 TaskTracker
hduser@hadoop:/usr/local$
```

Figure 5.1: Configuration of hadoop

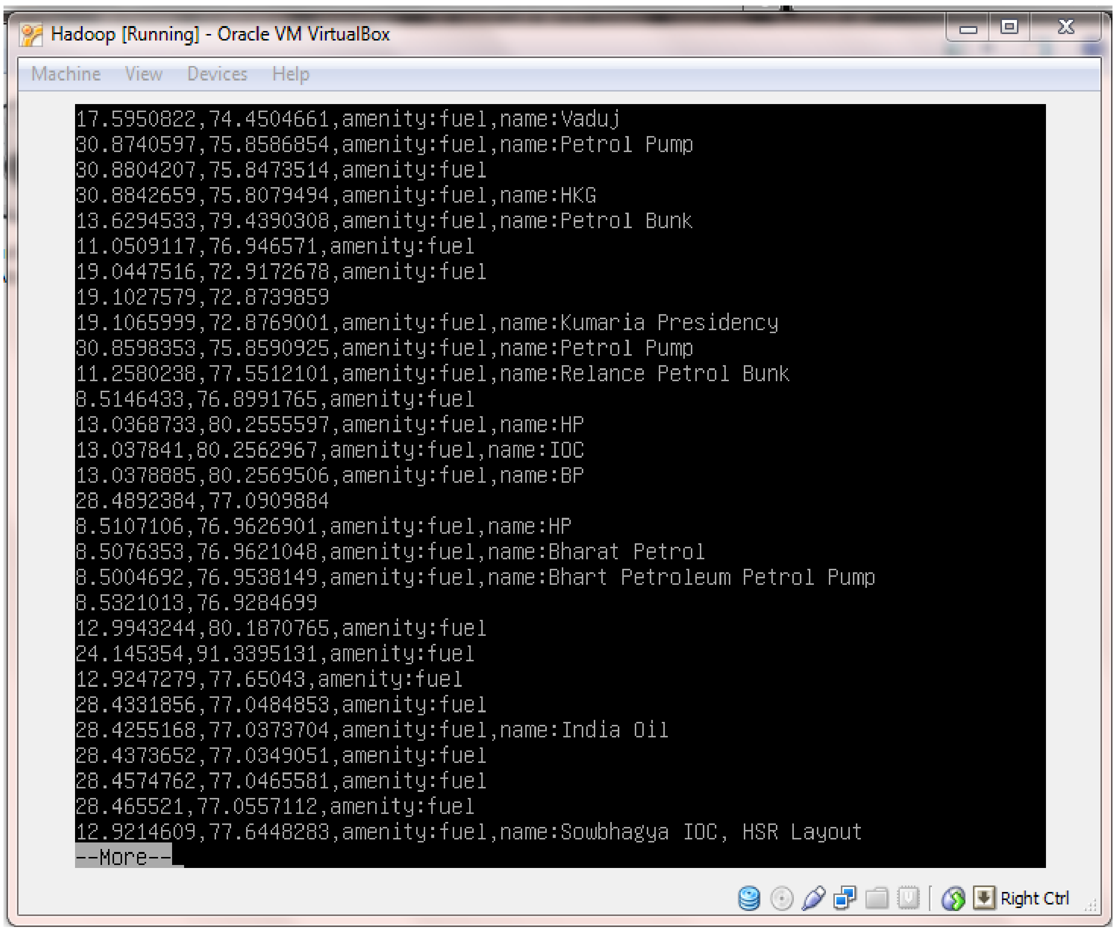
Figure 5.1 shows the pre map reduce processing. HDFS is used for the implementation of map reduce .So once the hadoop is installed, it is started by using the command “**start-all.sh**”. After this command hadoop is started on the linux. This is confirmed by using command “**jps**”. This command shows that whether all the nodes are up and working. So here it shows all the nodes to be working.


Hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
Right Ctrl

```
hduser@hadoop:~$ cd data
hduser@hadoop:~/data$ ls
atm.json  bar.txt          cafe.osm   hospital.osm  pub.osm          source
atm.osm   bus_station.json cafe.txt   library.json  pub.txt          tmp
atm.txt   bus_station.osm  fuel.json  library.osm   restaurant.json
bar.json  bus_station.txt  fuel.osm   library.txt   restaurant.osm
bar.osm   cafe.json        fuel.txt   pub.json      restaurant.txt
hduser@hadoop:~/data$
```

Figure 5.2: Categories of input files

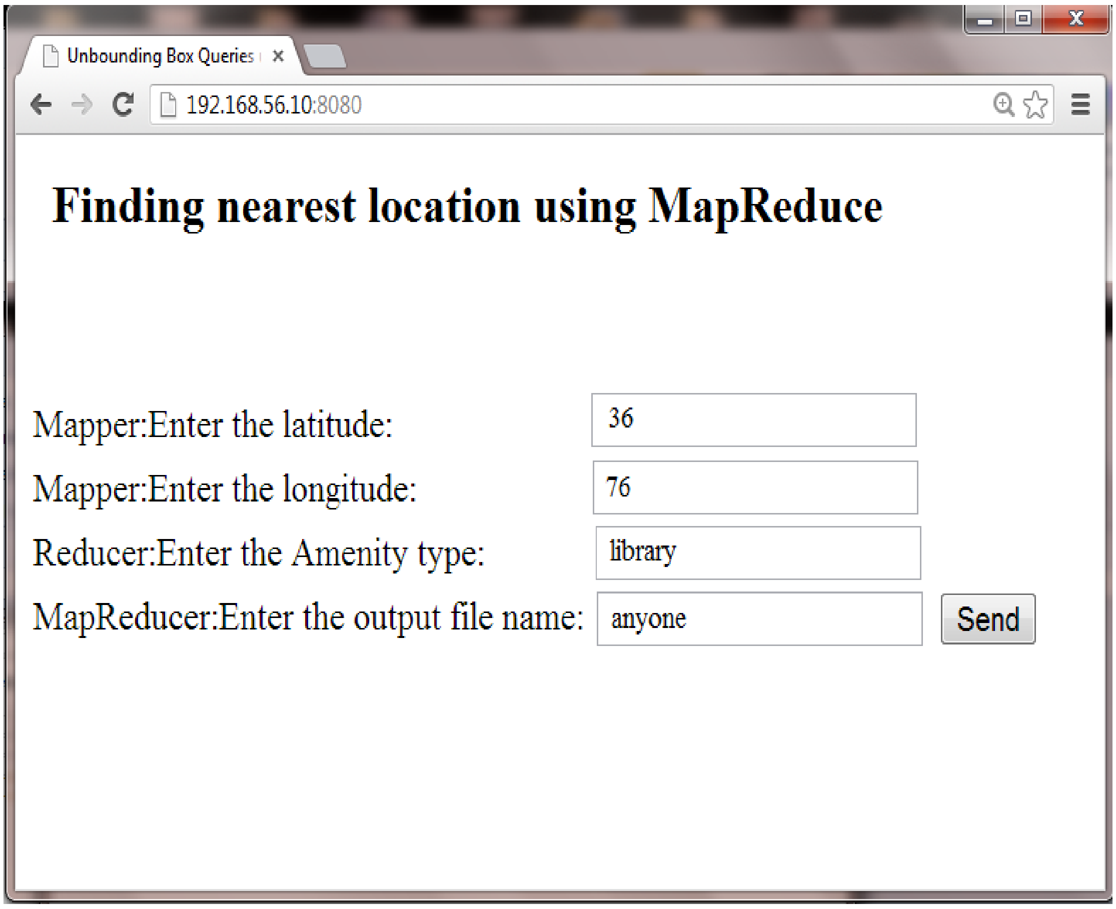
Figure 5.2 shows various categories of the locations that are being used here. One of these categories is needed to provide as a input further in order to find the nearest location of this category. There is a separate file created for every category. Every respective file contains the information about all the locations that are of this category. So more and more categories can be added further without tempering the old ones. Initially the files were present in the osm format. The osm format was converted in to json format and further this json format is converted in txt format and all the data is taken from txt format as it would be easy to read the data from this format.


Hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
Right Ctrl

```
17.5950822,74.4504661,amenity:fuel,name:Vaduj
30.8740597,75.8586854,amenity:fuel,name:Petrol Pump
30.8804207,75.8473514,amenity:fuel
30.8842659,75.8079494,amenity:fuel,name:HKG
13.6294533,79.4390308,amenity:fuel,name:Petrol Bunk
11.0509117,76.946571,amenity:fuel
19.0447516,72.9172678,amenity:fuel
19.1027579,72.8739859
19.1065999,72.8769001,amenity:fuel,name:Kumaria Presidency
30.8598353,75.8590925,amenity:fuel,name:Petrol Pump
11.2580238,77.5512101,amenity:fuel,name:Relance Petrol Bunk
8.5146433,76.8991765,amenity:fuel
13.0368733,80.2555597,amenity:fuel,name:HP
13.037841,80.2562967,amenity:fuel,name:IOC
13.0378885,80.2569506,amenity:fuel,name:BP
28.4892384,77.0909884
8.5107106,76.9626901,amenity:fuel,name:HP
8.5076353,76.9621048,amenity:fuel,name:Bharat Petrol
8.5004692,76.9538149,amenity:fuel,name:Bhart Petroleum Petrol Pump
8.5321013,76.9284699
12.9943244,80.1870765,amenity:fuel
24.145354,91.3395131,amenity:fuel
12.9247279,77.65043,amenity:fuel
28.4331856,77.0484853,amenity:fuel
28.4255168,77.0373704,amenity:fuel,name:India Oil
28.4373652,77.0349051,amenity:fuel
28.4574762,77.0465581,amenity:fuel
28.465521,77.0557112,amenity:fuel
12.9214609,77.6448283,amenity:fuel,name:Sowbhagya IOC, HSR Layout
--More--
```

Figure 5.3: Input file

Figure 5.3 shows the input file where all the data is saved which is used by the mapper. There has been separate datafile made for every category of place. This figure shows the datafile of category “fuel”. All the locations included in this file are of type fuel. Each line shows a different entry. First of all it shows longitude and latitude of a location. After that type of location is given, like here type of every location is fuel only. After that name of the location is given, every location has a different name. These are the longitudes and latitudes which are taken by mapper and are converted into the respective geohash code. Type is used because user has to give “type” of place he want to search for the nearest. Names are used to display as the time of output.



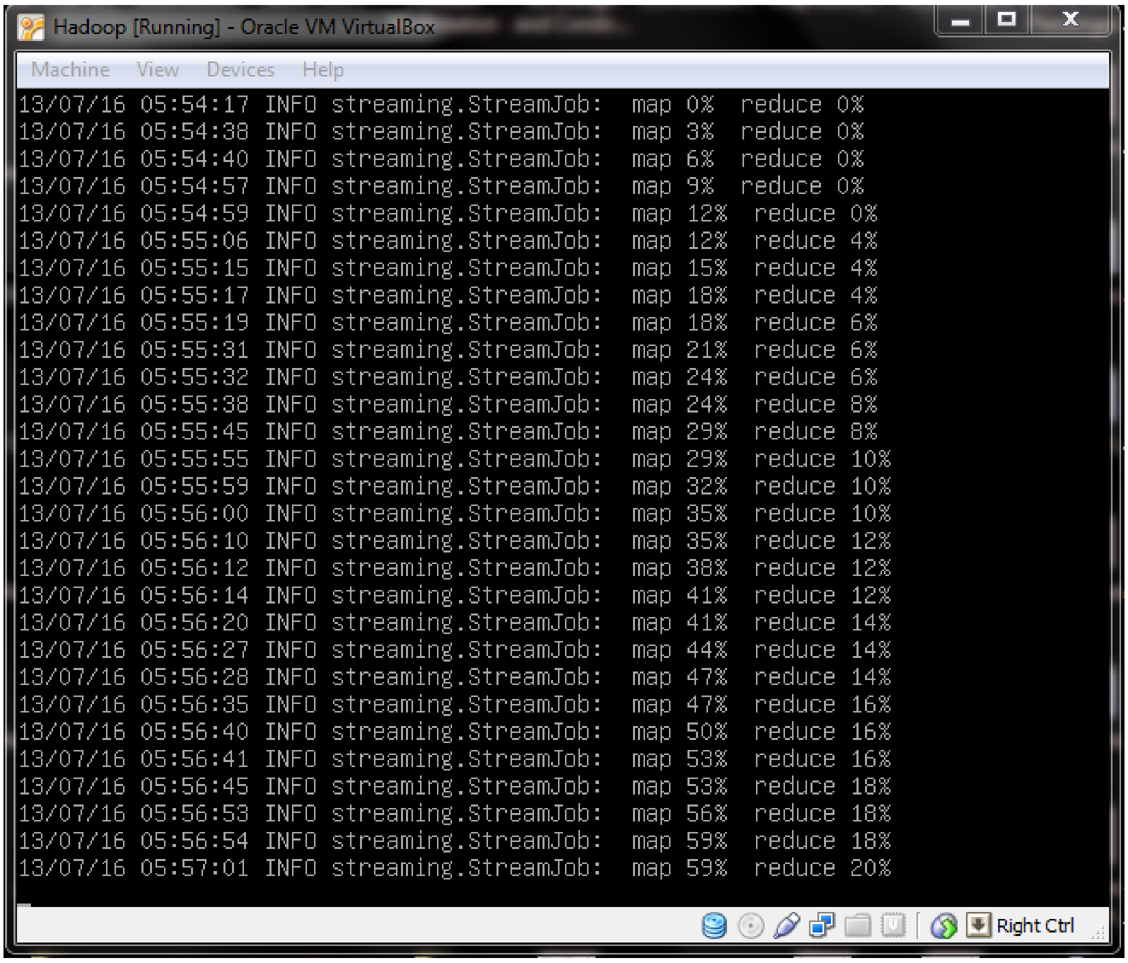
Finding nearest location using MapReduce

Mapper:Enter the latitude:	36
Mapper:Enter the longitude:	76
Reducer:Enter the Amenity type:	library
MapReducer:Enter the output file name:	anyone

Figure 5.4: Interface to find nearest location

Figure 5.4 shows the interface to the implementation. There are four columns in which options have to be provided by the user. The first column is for latitude which the user will put and second column is for longitude put by the user. These latitude and longitude are of the locations from where the user wants to find the nearest location of his choice. Third column is for type f the location. This depicts the type of place which the user wants to find nearest to him. Like if he want to find the nearest libraries, then he will put the libraries. He can put any type of location whose files are included are included as datafiles. And finally there is fourth column which in which user has to put the name of directory in which he wants his output file to be generated.

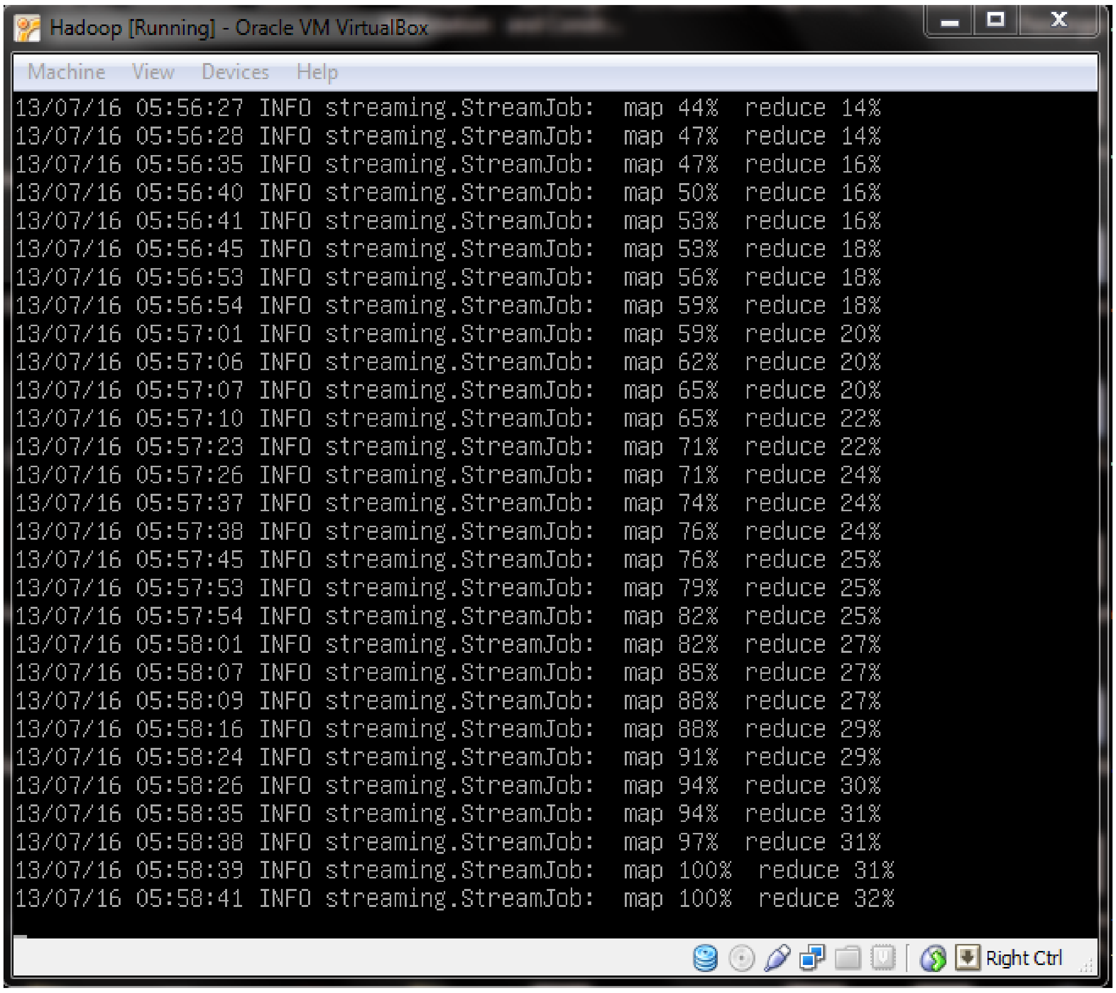
Column one and two are used by the mapper because geohash code is made using these and column two is used by the reducer which does the filtering based on type of place.


Hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
Right Ctrl

```
13/07/16 05:54:17 INFO streaming.StreamJob:  map 0%  reduce 0%
13/07/16 05:54:38 INFO streaming.StreamJob:  map 3%  reduce 0%
13/07/16 05:54:40 INFO streaming.StreamJob:  map 6%  reduce 0%
13/07/16 05:54:57 INFO streaming.StreamJob:  map 9%  reduce 0%
13/07/16 05:54:59 INFO streaming.StreamJob:  map 12%  reduce 0%
13/07/16 05:55:06 INFO streaming.StreamJob:  map 12%  reduce 4%
13/07/16 05:55:15 INFO streaming.StreamJob:  map 15%  reduce 4%
13/07/16 05:55:17 INFO streaming.StreamJob:  map 18%  reduce 4%
13/07/16 05:55:19 INFO streaming.StreamJob:  map 18%  reduce 6%
13/07/16 05:55:31 INFO streaming.StreamJob:  map 21%  reduce 6%
13/07/16 05:55:32 INFO streaming.StreamJob:  map 24%  reduce 6%
13/07/16 05:55:38 INFO streaming.StreamJob:  map 24%  reduce 8%
13/07/16 05:55:45 INFO streaming.StreamJob:  map 29%  reduce 8%
13/07/16 05:55:55 INFO streaming.StreamJob:  map 29%  reduce 10%
13/07/16 05:55:59 INFO streaming.StreamJob:  map 32%  reduce 10%
13/07/16 05:56:00 INFO streaming.StreamJob:  map 35%  reduce 10%
13/07/16 05:56:10 INFO streaming.StreamJob:  map 35%  reduce 12%
13/07/16 05:56:12 INFO streaming.StreamJob:  map 38%  reduce 12%
13/07/16 05:56:14 INFO streaming.StreamJob:  map 41%  reduce 12%
13/07/16 05:56:20 INFO streaming.StreamJob:  map 41%  reduce 14%
13/07/16 05:56:27 INFO streaming.StreamJob:  map 44%  reduce 14%
13/07/16 05:56:28 INFO streaming.StreamJob:  map 47%  reduce 14%
13/07/16 05:56:35 INFO streaming.StreamJob:  map 47%  reduce 16%
13/07/16 05:56:40 INFO streaming.StreamJob:  map 50%  reduce 16%
13/07/16 05:56:41 INFO streaming.StreamJob:  map 53%  reduce 16%
13/07/16 05:56:45 INFO streaming.StreamJob:  map 53%  reduce 18%
13/07/16 05:56:53 INFO streaming.StreamJob:  map 56%  reduce 18%
13/07/16 05:56:54 INFO streaming.StreamJob:  map 59%  reduce 18%
13/07/16 05:57:01 INFO streaming.StreamJob:  map 59%  reduce 20%
```

Figure 5.5: Initial stage of mapping and reduction

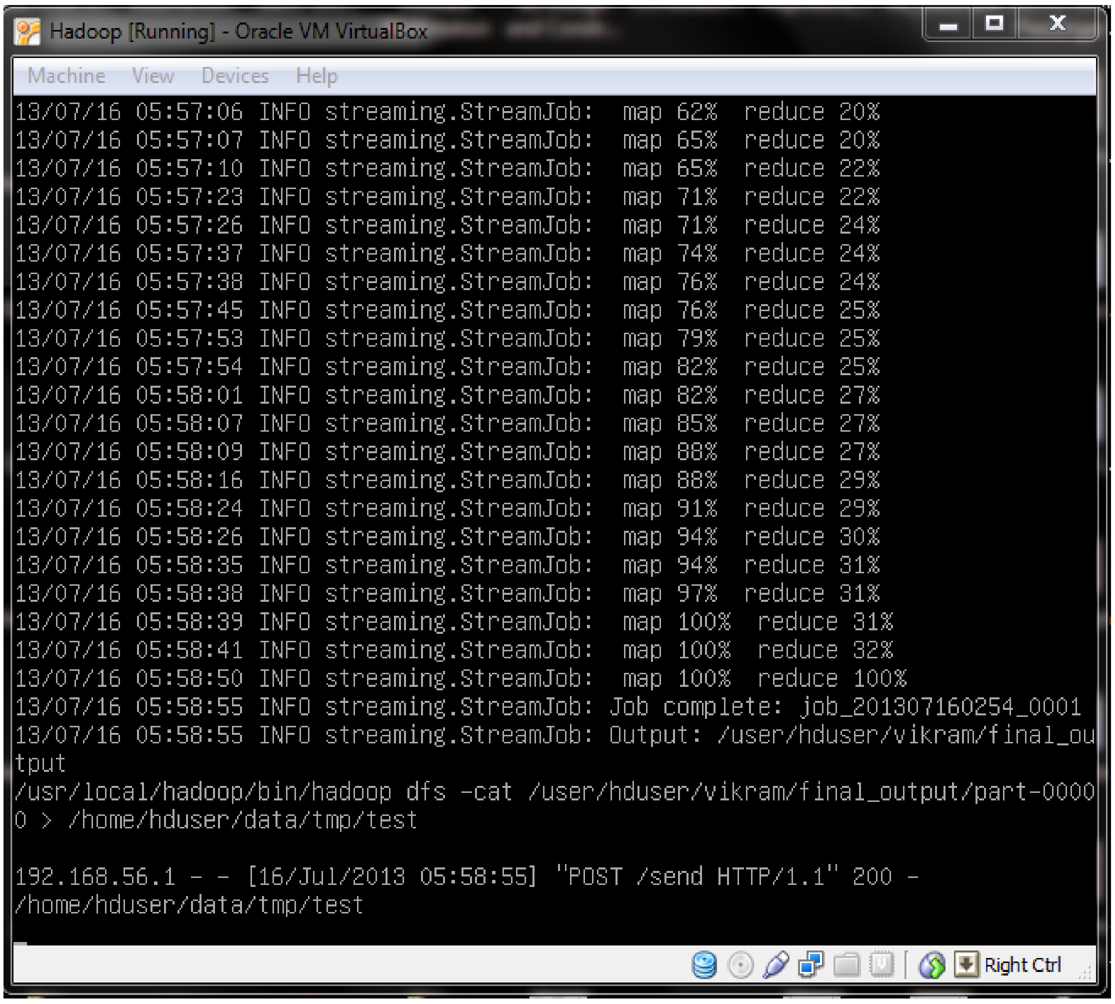
Figure 5.5 shows the first phase when MapReduce job has started. In the starting both mapping and reduction are 0 percent. It can be seen that they both start simultaneously the data which has been mapped by the mapper, reducer starts to reduce it simultaneously. It means the input values which have been converted by mapper in to the corresponding geohash code are accepted by the reducer simultaneously.


Hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
Right Ctrl

```
13/07/16 05:56:27 INFO streaming.StreamJob:  map 44%  reduce 14%
13/07/16 05:56:28 INFO streaming.StreamJob:  map 47%  reduce 14%
13/07/16 05:56:35 INFO streaming.StreamJob:  map 47%  reduce 16%
13/07/16 05:56:40 INFO streaming.StreamJob:  map 50%  reduce 16%
13/07/16 05:56:41 INFO streaming.StreamJob:  map 53%  reduce 16%
13/07/16 05:56:45 INFO streaming.StreamJob:  map 53%  reduce 18%
13/07/16 05:56:53 INFO streaming.StreamJob:  map 56%  reduce 18%
13/07/16 05:56:54 INFO streaming.StreamJob:  map 59%  reduce 18%
13/07/16 05:57:01 INFO streaming.StreamJob:  map 59%  reduce 20%
13/07/16 05:57:06 INFO streaming.StreamJob:  map 62%  reduce 20%
13/07/16 05:57:07 INFO streaming.StreamJob:  map 65%  reduce 20%
13/07/16 05:57:10 INFO streaming.StreamJob:  map 65%  reduce 22%
13/07/16 05:57:23 INFO streaming.StreamJob:  map 71%  reduce 22%
13/07/16 05:57:26 INFO streaming.StreamJob:  map 71%  reduce 24%
13/07/16 05:57:37 INFO streaming.StreamJob:  map 74%  reduce 24%
13/07/16 05:57:38 INFO streaming.StreamJob:  map 76%  reduce 24%
13/07/16 05:57:45 INFO streaming.StreamJob:  map 76%  reduce 25%
13/07/16 05:57:53 INFO streaming.StreamJob:  map 79%  reduce 25%
13/07/16 05:57:54 INFO streaming.StreamJob:  map 82%  reduce 25%
13/07/16 05:58:01 INFO streaming.StreamJob:  map 82%  reduce 27%
13/07/16 05:58:07 INFO streaming.StreamJob:  map 85%  reduce 27%
13/07/16 05:58:09 INFO streaming.StreamJob:  map 88%  reduce 27%
13/07/16 05:58:16 INFO streaming.StreamJob:  map 88%  reduce 29%
13/07/16 05:58:24 INFO streaming.StreamJob:  map 91%  reduce 29%
13/07/16 05:58:26 INFO streaming.StreamJob:  map 94%  reduce 30%
13/07/16 05:58:35 INFO streaming.StreamJob:  map 94%  reduce 31%
13/07/16 05:58:38 INFO streaming.StreamJob:  map 97%  reduce 31%
13/07/16 05:58:39 INFO streaming.StreamJob:  map 100%  reduce 31%
13/07/16 05:58:41 INFO streaming.StreamJob:  map 100%  reduce 32%
```

Figure 5.6: Middle stage of mapping and reduction

Figure 5.6 shows that mapping has been completed to 100 percent and reduction is still going on and has been completed up to 32 percent. It means all the input entries have been converted to the corresponding geohash code. And reducer has started to filter them now. Reducer will output only those locations whose type matched the type defined by the user.

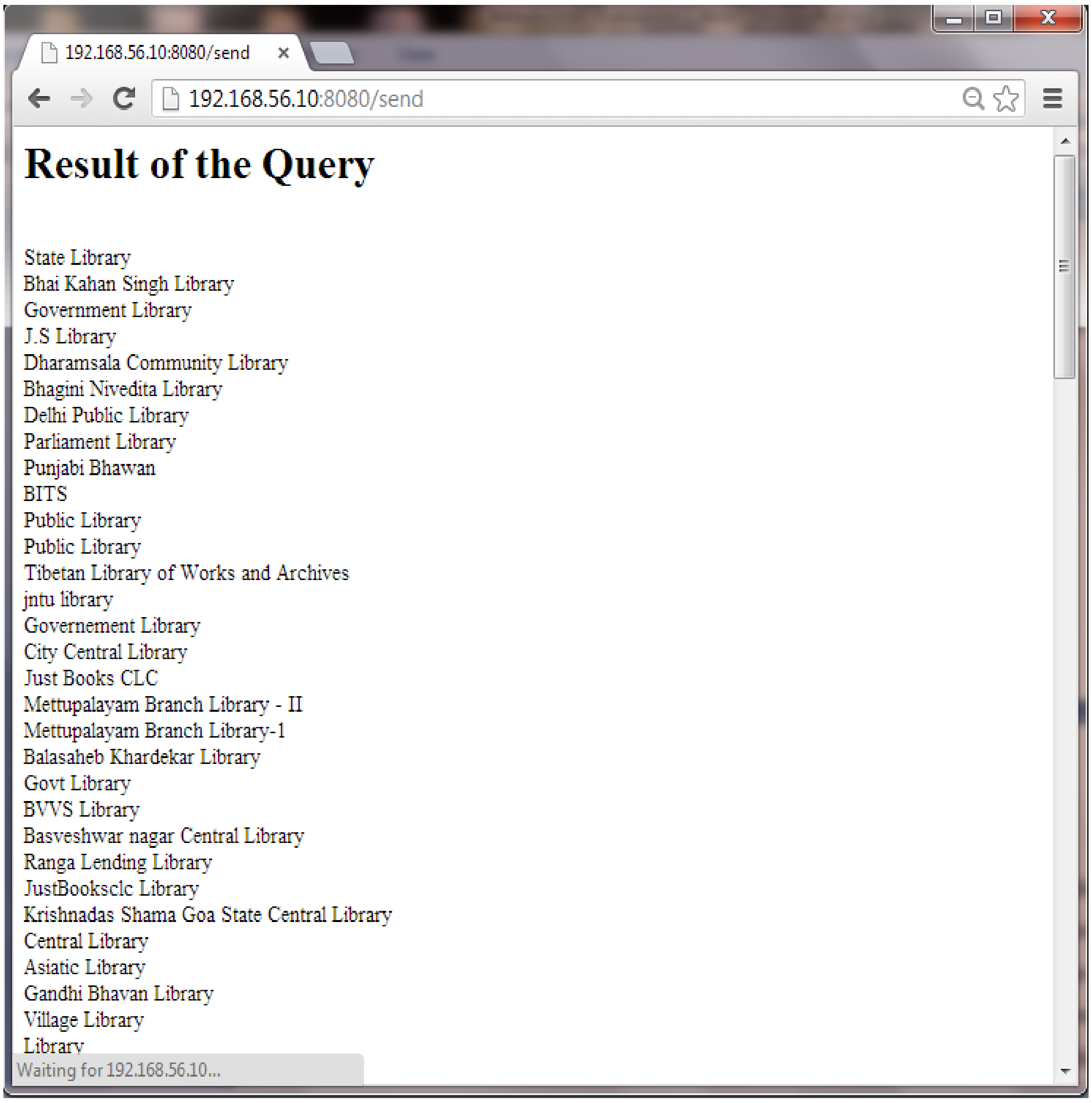

Hadoop [Running] - Oracle VM VirtualBox
Machine View Devices Help
Right Ctrl

```
13/07/16 05:57:06 INFO streaming.StreamJob:  map 62%  reduce 20%
13/07/16 05:57:07 INFO streaming.StreamJob:  map 65%  reduce 20%
13/07/16 05:57:10 INFO streaming.StreamJob:  map 65%  reduce 22%
13/07/16 05:57:23 INFO streaming.StreamJob:  map 71%  reduce 22%
13/07/16 05:57:26 INFO streaming.StreamJob:  map 71%  reduce 24%
13/07/16 05:57:37 INFO streaming.StreamJob:  map 74%  reduce 24%
13/07/16 05:57:38 INFO streaming.StreamJob:  map 76%  reduce 24%
13/07/16 05:57:45 INFO streaming.StreamJob:  map 76%  reduce 25%
13/07/16 05:57:53 INFO streaming.StreamJob:  map 79%  reduce 25%
13/07/16 05:57:54 INFO streaming.StreamJob:  map 82%  reduce 25%
13/07/16 05:58:01 INFO streaming.StreamJob:  map 82%  reduce 27%
13/07/16 05:58:07 INFO streaming.StreamJob:  map 85%  reduce 27%
13/07/16 05:58:09 INFO streaming.StreamJob:  map 88%  reduce 27%
13/07/16 05:58:16 INFO streaming.StreamJob:  map 88%  reduce 29%
13/07/16 05:58:24 INFO streaming.StreamJob:  map 91%  reduce 29%
13/07/16 05:58:26 INFO streaming.StreamJob:  map 94%  reduce 30%
13/07/16 05:58:35 INFO streaming.StreamJob:  map 94%  reduce 31%
13/07/16 05:58:38 INFO streaming.StreamJob:  map 97%  reduce 31%
13/07/16 05:58:39 INFO streaming.StreamJob:  map 100%  reduce 31%
13/07/16 05:58:41 INFO streaming.StreamJob:  map 100%  reduce 32%
13/07/16 05:58:50 INFO streaming.StreamJob:  map 100%  reduce 100%
13/07/16 05:58:55 INFO streaming.StreamJob: Job complete: job_201307160254_0001
13/07/16 05:58:55 INFO streaming.StreamJob: Output: /user/hduser/vikram/final_ou
tput
/usr/local/hadoop/bin/hadoop dfs -cat /user/hduser/vikram/final_output/part-0000
0 > /home/hduser/data/tmp/test

192.168.56.1 - - [16/Jul/2013 05:58:55] "POST /send HTTP/1.1" 200 -
/home/hduser/data/tmp/test
_
```

Figure 5.7: Final stage of mapping and reduction

Figure 5.7 shows the final phase of the MapReduce job. Both the Map and reduce jobs have been completed, there showing to be 100 percent completed. And the output file has been generated in the directory which was specified.



Result of the Query

```
State Library
Bhai Kahan Singh Library
Government Library
J.S Library
Dharamsala Community Library
Bhagini Nivedita Library
Delhi Public Library
Parliament Library
Punjabi Bhawan
BITS
Public Library
Public Library
Tibetan Library of Works and Archives
jntu library
Governement Library
City Central Library
Just Books CLC
Mettupalayam Branch Library - II
Mettupalayam Branch Library-1
Balasaheb Khardekar Library
Govt Library
BVVS Library
Basveshwar nagar Central Library
Ranga Lending Library
JustBooksclc Library
Krishnadas Shama Goa State Central Library
Central Library
Asiatic Library
Gandhi Bhavan Library
Village Library
Library
```

Waiting for 192.168.56.10...

Figure 5.8: Output

Figure 5.8 shows the output file. In the output file, names of all the libraries have been displayed. They are displayed in the order of length of prefix matched. It means the name at the top is nearest to the location that was entered and name at the bottom is farthest from the entered location. Only libraries have been displayed here. So they are in the order of increasing farness.

Chapter 6
Conclusion and Future Scope

6.1 Conclusion

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. A lot of frameworks and techniques are there to handle geospatial queries. In this thesis a method to tackle nearest location problem has been derived. There are already existing methods to find the nearest neighbor but they were somehow not so accurate and efficient according to this problem. So a more refined method has been presented which uses longitude and latitude of the locations.

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 parallelly. It improves the solution in terms of time complexity to a great extend because the data is being processed parallel. A variety of geospatial queries can be efficiently modeled using MapReduce. MapReduce is a programming model that lets developers focus on the writing 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. There MapReduce is very well suited for the Geospatial queries.

6.2 Future Scope

Geospatial queries play an important role in modern day where geospatial data is growing rapidly. And finding the nearest location is one of the most important geospatial query. 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 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 find the nearest location around it where it can start its service but at same time has adequate population and similarly for other organisations.

Bibliography

[1] Pascal Hitzler and Krzysztof Janowicz “Linked Data, Big Data, and the 4th Paradigm*” IOS Press and the authors Semantic* Web 4,2013

[2] Nathan Eagle “Big data, global development, and complex social systems”,*ACM New York, NY, USA* 2010

[3] Wenfei Fan “Querying Big Social Data” *Springer Berlin Heidelberg BNCOD* 2013

[4] Marissa Mayer “The Coming Data Explosion” *Richard MacManus* cited as http://www.readwriteweb.com/archives/the_coming_data_explosion.ph,May 30, 2010

[5] Alex Nazaruk,Michael Rauchman “Big Data in Capital Markets” *SIGMOD*’13, June 22–27, 2013, New York

[6] Afsin Akdogan, Ugur Demiryurek, Farnoush Banaei Kashani, Cyrus Shahabi “Voronoi-Based Geospatial Query Processing with MapReduce*” CloudCom* Pg. 9-16, 2010

[7] J.M. Patel “Building a Scalable Geospatial Database System” *In SIGMOD,* 1997

[8] “Geohash and its format” [online] http://geohash.org/site/tips.html

[9] Jens Dittrich, JorgeArnulfo Quian´eRuiz “Efficient Big Data Processing in Hadoop MapReduce” *Proceedings of the VLDB Endowment, Vol. 5, No*. 12, August 27th 31st 2012, Istanbul, Turkey.

[10] Kyuseok Shim “MapReduce Algorithms for Big Data Analysis” Proceedings of the *VLDB Endowment, Vol. 5, No. 12,* 2012

[11] Wei Pan, Zhanhuai Li,Qun Chen,Shanglian Peng,Bo Suo,Jian Xu “MSMapper: An Adaptive Split Assignment Scheme for MapReduce” *Springer Berlin Heidelberg,*2012

[12] Lizhe Wang,Jie Tao,Holger Marten,Achim Streit,Samee U. Khan,Joanna Kolodziej,Dan Chen “MapReduce across Distributed Clusters for Data-intensive Applications*”, IEEE* 2012

[13] Jeffrey Dean, Sanjay Ghemawat "MapReduce: simplified data processing on large clusters." *Commun. ACM Vol. 51*,2008

[14] Hui Jin, Kan Qiao, Xian-He Sun, Ying Li "Performance under Failures of MapReduce Applications*"* *CCGRID Pg. 608-609* , 2011

[15] "Hadoop fundamentals" [online] http://bigdatauniversity.com/courses/

[16] "What is hadoop" [online] http://hadoop.apache.org/

[17] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung "The Google file system" *SOSP* Pg. 29-43, 2003

[18] Hieu Hanh Le, Satoshi Hikida, Haruo Yokota "NameNode and DataNode Coupling for a Power-Proportional Hadoop Distributed File System" *DASFAA* (2) Pg. 99-107, 2013

[19] D. Jiang, B. Chin, O. L. Shi, and S. Wu. "The Performance of MapReduce: An In-depth Study", *VLDB*, 2010.

[20] T. Bleier and F. Freund. "Earthquake warning system*."* *Spectrum, IEEE*, 2005.

[21] K. Borau, C. Ullrich, J. Feng, and R. Shen. "Microblogging for language learning: Using twitter to train communicative and cultural competence". *In Proc. ICWL 2009*, pages 78–87, 2009

[22] Wichian Premchaiswadi, Walisa Romsaiyud "Optimizing and Tuning MapReduce Jobs to Improve the Large-Scale Data Analysis Process." *Int. J. Intell. Syst.* Vol. 28 No. 2 Pg. 185-200, 2013

[23] David Jiang, Anthony K. H. Tung, Gang Chen "MAP-JOIN-REDUCE: Toward Scalable and Efficient Data Analysis on Large Clusters" *IEEE Trans. Knowl. Data Eng.* Vol. 23 No. 9 Pg. 1299-1311 , 2011

[24] Berners-Lee T, Hendler J, Lassila O. "The semantic web", *Scientific American* May 2001

[25] "W3C recommendation: Rdf primer" http://www.w3.org/TR/rdf-primer/

[26] Jacopo Urbani, Jason Maassen, Niels Drost, Frank J. Seinstra, Henri E. Bal " Scalable RDF data compression with MapReduce" *Concurrency and Computation: Practice and Experience* Vol. 25 No. 1 Pg. 24-39, 2013

[27] Der-Tsai Lee “On -Nearest Neighbor Voronoi Diagrams in the Plane” *IEEE Trans. Computers*, 1982

[28] Tomoyuki Shibata, Toshikazu Wada “K-D Decision Tree: An Accelerated and Memory Efficient Nearest Neighbor Classifier” *IEICE Transactions* Vol. 93-D No. 7 Pg. 1670-1681, 2010