Safe Route Recommender : An efficient way to find safe and short path between source and destination

Pratik Bhangale
Computer Science, UMBC
Email: pratikb1@umbc.edu

Vishal Rathod
Computer Science, UMBC
Email: vishalr1@umbc.edu

Mayur Pate
Computer Science, UMBC
Email: mayurp1@umbc.edu

***Abstract*—Over the last decade , the use of maps has increased dramatically. Every car or traveling individual use Google maps for navigating to their desired destination. Although the routes provided by Google maps helps users to figure which is the shortest or traffic free path, but here safety is not taken into consideration. In this paper, we propose and develop a mobile application that utilizes crime data to recommend the safe and shortest route between source and destination. We model the calculation of the safeness of the route based on 3 different data sets a) Crime records b) Accident cases c) Police station and health care services. Specifically, Using data of Baltimore city we develop a safeness model for street urban network which allows us to give safe factor to every street. The model uses hadoop and spark scripts for big data processing. Furthermore the paper also discusses caching system used to provide real time response to the mobile app and also identifies what could be done to improve such systems.**

***Keywords*—Navigation, Big data, Android, NoSql, Hadoop, Spark, Web service, Google Maps, REST API.**

I. INTRODUCTION

Today, 54 per cent of the worlds population lives in urban areas, a proportion that is expected to increase to 66 per cent by 2050. Projections show that urbanization combined with the overall growth of the worlds population could add another 2.5 billion people to urban populations by 2050 [1] says un.org. One of the main reasons for these levels of urbanization is the long-lived thought of cities as the paramount instrument for innovation and wealth creation. Due to increasing population density in cities, they have become the main source of crimes, diseases and pollution, which is significantly deteriorating the quality of life in cities. We have concentrated on the data of Baltimore city. According to the FBI (Uniform crime report) Baltimore, Maryland is one of the high crime rated city[2]. Many government authorities are maintaining a lot of data about the crime and accidents. The open government and data initiative from President Obamas administration [3] has further led many local authorities to systematically collect, organize and publicize such datasets. In this paper, we focus on finding a solution for one of the major aforementioned problems present in almost every megacity today: crimes and public safety. We develop a mobile application and big data system aiming to identify safe and short paths for people to take when they navigate through the city. For this purpose we analyze both crime and accident cases present on a particular path. Major challenge in this project is to approximate between safe and shortest path. So ideally, prioritizing the safety of dwellers, one would like to provide the user with the safest path from origin x to destination y. We have developed a mobile application where user will enter the source and destination and user will be able to see the multiple paths according to their safe scores on the map interface in app. The paper is structured as follows: Section 2 provides a motivation for developing this safe route recommender service. Section 3 gives an overview of related works done till date and how our idea is unique for commuters. Section 4 presents details about who will be actual users of this service, and then approach and implementation details of this services, we discuss the algorithmic approach and big data computations to get the intended results and how system can be scaled to handle large amounts of requests. The paper concludes with a summary and results in Section 10 and onward.

II. MOTIVATION

The way we navigate in cities has been revolutionized in the last few years by the advent of

GPS mapping programs. Most GPS mapping apps are designed to get you where you want to go quickly just by entering your start and end location and these will give you the shortest route from A to B. But consider if a tourist or a traveller is new two a certain place and he doesn't aware that how secure is that place for travelling through vehicles or as a pedestrian. If considered places like Detroit, where a violent crime rate of 1,988.63 per 100,000 people, similarly for Memphis its 1,740.51, Oakland 1,685.39 and for Baltimore its 1,338.54, etc. Some of the most prominent crimes in the city are aggravated assault, assault with a deadly weapon, rape, armed robbery, murder and non-negligent manslaughter. Along with such records there are also places where its too unsecure for vehicular travelling due to road conditions and structures which caused tremendous increases in accidental cases in last few years. These all cases which makes a reasonable requirement of alternative paths those are not only short by distance but also secure for travelling or walking.

So tackle the everyday problem of public i.e. google map helps to find the shortest route with traffic information etc but it sometimes shows the shortest path that involves a deserted route; which is not safe. Thus the motivation chosen is an apt one to develop a service which is useful for every individual today for whom the safest routes to a destination, which is also an important factor. This service plays vital role for those who unaware of a particular area/city. Also this service can be embedded with apps widely used like Ways, Google Maps. The project deals with a real world huge openly available data and addresses a social problem of making the travel of people more convenient and safe. The designed algorithms of this project also supports those commuters are often concerned with the quickest path to a destination by recommending the trade of between short and safe path. The goal of this work is to automatically suggest routes that are not only short but also secure.

III. Related Works

There are few developments on the similar lines of the system we have built, some of which are enlisted below:

- The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City. The Yahoo researchers have developed the algorithm which allows user to take the Scenic Route through a new Mapping Software which gives 'Beauty Scores of different routes between source and destination. The goal of this work is to automatically suggest routes that are not only short but also emotionally pleasant, they say. They then crowd sourced opinions about the beauty of each location using a website called UrbanGems.org.

- Safe Route - San Francisco 2016 TechCrunch Disrupt Hackathon. The one of runners-up team built an application SafeRoute, a hack designed to give users a safer path to their destination. As its name implies, Safe Route is designed to give users a safer path to their destination. The app utilizes Google Maps, Google Places, and Crimemapping.com APIs to help determine the best way to get from point A to point B, even if it means making the route a bit longer in the process.

- Safe Navigation in Urban Environments: This is a application that utilizes crime data to provide safe urban navigation. Specifically, using crime data from Chicago and Philadelphia, its a developed risk model for their street urban network which allows us to estimate the relative probability of a crime on any road segment.

IV. Service Users

The service is useful for every individual today who travels with the help of mobile maps.Its very useful those who is new at a particular area/city. All the everyday commuters such as taxi drivers, pedestrian and tourists are the users to whom this service will provide the safest and shortest route. Apart from this it might help government to employ more security protocols on the routes those are marked unsafe by this service.

V. Datasets and Formatting

We collected crime and accident data sets from data.gov website. Data.gov is the official website of

US Governments for open data. We collected data sets for various cities like New York, San Francisco and Baltimore. For implementation purposes, we decided to stick with only one city. We choose Baltimore city and used all its crime and accident data during implementation of this project. We collected crime data which had 27000 crimes over the last 2 years. We have also collected accident data which has more than 5000 records for last 2 years alone in Baltimore. We collected all this data in well-structured CSV files. We also need to collect data for police and health care service in Baltimore city. We collected this data using Google Places API. We implemented this API and dumped all data in CSV file which contains almost 1000 entries for police and healthcare services.

Though all CSV files were well-structured, we had to change the structure of CSV files to make spark computations faster. These files stored latitude and longitude in single string format. For example, (12.33223 -12.2323). We needed these strings to be separated into different columns. So, we wrote a python script which reads input CSV file and converts it to custom structure specified by user. We performed this cleaning operation for both crime and accident records.

VI. Approach

Here's how we planned it to build :

An aim it to build a product which will allow an user to enter a current or start location, along with a desired destination and then by algorithmic processing it will select and suggest the route that ranks highest on a "safe score" attributed to each route by the software. Safe route recommender consists of three main parts: one that is responsible for computing the safe score of routes through big data computations, another part is find trade of between short and safe route using several algorithms and lastly to represents the different routes according to the results computed and recommend the route neatly on friendly user interface.

The algorithm techniques to generate the safe score of every path between source and destination consumes the input information as set of datasets mentioned above. These data sets have been collected from the authenticated government sources. And processing these information it will help to determine the safeness of the routes. Additionally, we have used Google API's to get the different routes between source and destination. This API provides meta data about the paths as a response so it again helps to get the distances between two paths. So processing the API results by different algorithms, its possible to generate the trade off between short and safest path which ultimately helps commuters to have short and safest and wonderful travel. The service which we built that locates the user and allows to find him/her the efficient path between the current and destination location. We are using the big data computation technologies such as Hadoop, Spark for the algorithmic computations. To work out whether the routes chosen by the algorithm are really safe and short, we have considered the data of a particular location and verifying the results. We are showing the final results on the mobile application with user friendly map interface. We have also build a software technique which allows to handle large number of requests and those requests for which currently safe factor is not present in the system. Overall systems aims to provide the better user experience and recommend the safe path by processing authenticated data.

VII. Web Service Architecture

We have designed our system using three tier software architecture. We have developed an android application which is a presentation layer of our system. Our logic tier is responsible for request handling, algorithmic computations and response generation. Finally, data tier is divided into two main parts. First part consists of traditional NoSQL database which processes and returns all data in real time. Second part consists of Big data stored in Hadoop clusters. This data is used for safe score generation. Details of Safe score generation and data processing will be explained later in this paper.Our system provides an elegant interface to the users of this application. User provides a source and destination location as an input from android application. Then our android application makes a service call to application server by using REST protocol. This call is uses HTTP GET method for sending request parameter to server in form of JSON string. Using these input parameters, our

web service calls Googles maps API to fetch all possible routes between source and destination.

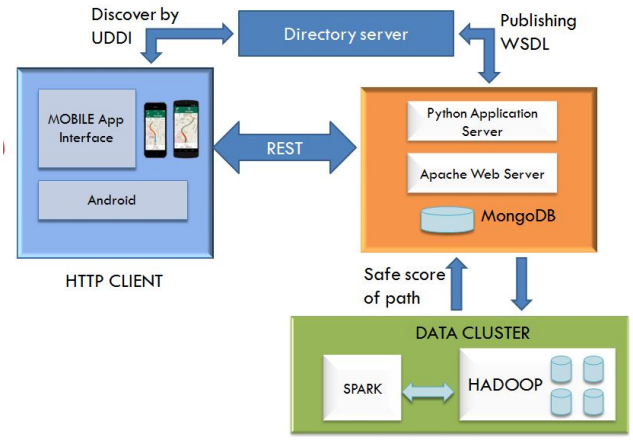

Fig. 1. System Architecture

Once we have all routes, we fetch safe score for each route from MongoDB and then we apply our approximation technique to suggest best possible routes for travel. Our service returns these paths in encoded polyline string in JSON format to android application. Then android application decodes these encoded polyline strings and convert each route to actual location co-ordinates for plotting them on map.

As mentioned previously, we have divided data layer into two main parts. First part consists of MongoDB - NoSQL database which we have included to process all real-time requests. Currently, MongoDB layer only stores route information and count for crimes and accidents happened on that route. Second part consists of Hadoop cluster and Spark. This layer stores crime, accident, healthcare and police services data. This is nothing but Big Data in our system. We are processing this big data using Spark scripts which are responsible for generating counts for every route that comes as request. We have designed caching approach by which both these layers communicate with each other. This caching approach will be explained later in this paper.

VIII. Technologies

We have used various technologies to develop this system. We will be listing all technologies per logical tiers.

A. Front End

We have developed an Android application as a front end. This application has used technologies such as Java8, Google maps API, Google Volley and material design etc. Google maps and Volley API has been used for plotting routes on maps and providing auto complete functionality for users input of source and destination locations.

B. Back End

We have developed our web service using Pylons Python web framework for developed service oriented applications. Pylons application are developed in Python language and runs with a utility called as Paster. We have also used pymongo database driver for communicating with MongoDB from pylons application. Additionally, we have used useful python libraries such as Google Maps server side API and hashlib. As mentioned previously, we have used NoSQL database MongoDB 2.4.9 for the development of this project. We have used inbuilt utility in MongoDB called as mongo client for development and debugging of our application.

C. Big Data

We have used Hadoop clusters to store data from various source such as crime and accident. We have used Hadoop 2.7 for development of this application. We stored structured data in csv file format in HDFS using PUT command provided in hadoop binaries. We have used pyspark on top of HDFS to fetch and process data from HDFS clusters.

D. Production Environment

We have deployed this application on public cloud using Amazon web services. We have used EC2 micro instance to host our application. We have also deployed our android application on Google play store and its publicly available in all countries in the world. This application currently runs on android mobiles and tablets. We have also configured single node cluster running Hadoop and spark jobs.

IX. Implementation Details

We have used modular approach for development of this project.

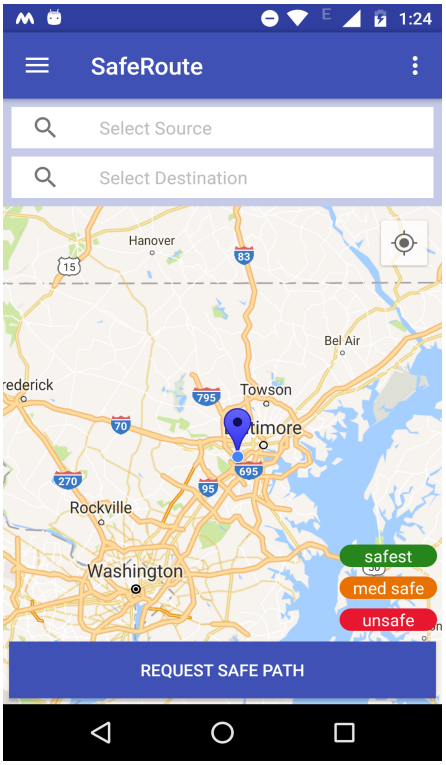

Fig. 2. User enters source and destination

In following points, we will be discussing development of every module in detail in terms of technical implementation and challenges.

A. User Interface

The project is mainly focused on map navigation so most handy interface was an mobile app. We have developed an android app - SafeT, for showing the safe routes to user. Here user enter source and destination. Figure 2 gives the screen shot of the user input screen.

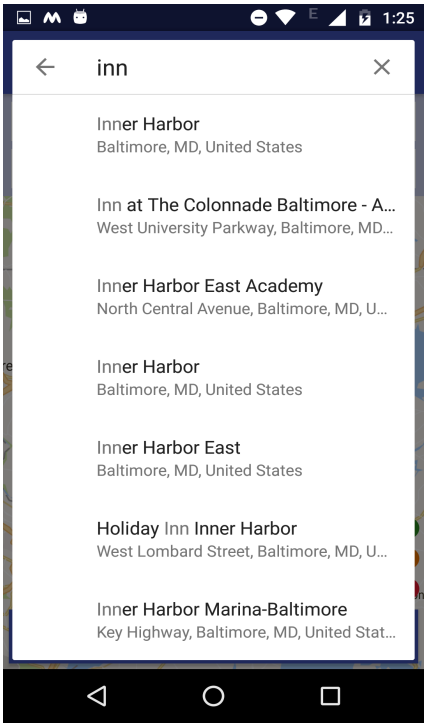

Fig. 3. Google places suggestions

We have integrated Google Place API to provide user with real places suggestion . This is show in figure 3. When the user enter source and destination and requests to see the route. The request is sent to the web server. The app then receives Json response of the routes which are in the form of polyline string and their respective safe scores.

We use Google Volley API for communicating with the server.The detailed format of response is explained in later section in paper. Polyline encoding is a lossy compression algorithm that allows you to store a series of coordinates as a single string [4]. Point coordinates are encoded using signed values. The app then decodes this polyline string to get a series of latlong points and plots them on the Google maps. The app uses google maps v2 API for this. According to safe score the app displays the routes as follows : Safest - green , Orange - medium safe , Red- unsafe the output is shown in figure 4.

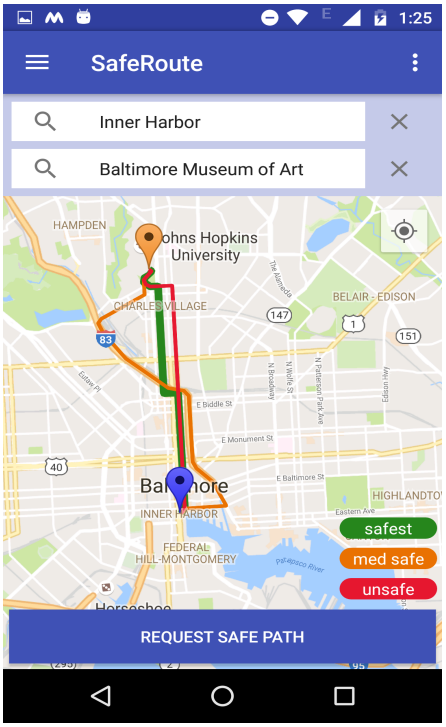

Fig. 4. Google places suggestions

B. Web Service Development

Android application passes the source and destination input in simple JSON string to web service. This web service reads these request parameters from URL and then makes a call to google maps server side API. Google provides a key based authentication mechanism for performing any type of API calls. Once the authentication has been done, google returns client stub authenticated token which can be used to make further calls for limited session. Once session expires or web service loses request instance, one must reauthenticate for making future API calls to google maps. After successful authentication, our web service passes source, destination and authenticated token to Google Directions API which in turn returns the huge JSON route array as output. Each entry of this array contains route information, polyline string and distance. Our web service extracts polyline string and distance for each route and stores this in temporary data structure. As explained in last point, polylines are encoded strings which stores series of latitude and longitude in smaller length string. Our web service further convert these polylines into secure hash of 32 bits using RSA md5 encryption technique. We have converted these polylines to hash keys as we must uniquely identify each route in our system. We could have used polyline strings as a unique identifier, but performing long length and string intensive queries in not recommended approach in MongoDB. So, we converted these polylines into manageable hash keys and stored them as primary key for each route in our database.

Our web service then makes a search in database with hash keys for every path to fetch crime count, accident count and healthcare services. These counts are generated from processing of crime and hospital data in Spark and then updated to MongoDB. How these counts are generated and for which routes its generated will be explained in further sub points. For simplicity, we will be assuming that we get some count value for crime and accident for every route searched with unique hash keys. Web service then applies weighted average algorithm on them to generate safe score levels for every path. This algorithm is explained below in detail. These algorithm considers both safety and shortness factor and makes a best possible sequence of routes for travel. Here challenge was to balance out both factors and make best choices. Details of this algorithm as follows:

C. Safe Score Generation Algorithmic Techniques

1) Weighted Average Algorithm: We have chosen a weighted average technique cause its more accurate measurement of scores or investments that are of relative importance to each other. It is very similar to an ordinary arithmetic mean (the most common type of average), except that instead of each of the data points contributing equally to the final average, some data points contribute more than others. So it can also be considered as priority based average algorithm. In case of recommending safe route, let see how this technique helps in determining the safe route among different routes between two locations.As mentioned in data sets section and shown in below figure we are using three types of data which are treated as a factor while computing the weighted average. On execution of sparks jobs the generated counts are multiplied by the decided factor or weight and after multiplication all three types of scores are added together which decides

the safe score of a path. The priorities or weightage of these factors are decided in percentages. While testing we have given the 50% weight to crime events, 30% weight to accidents and 20% to number of police and health services.

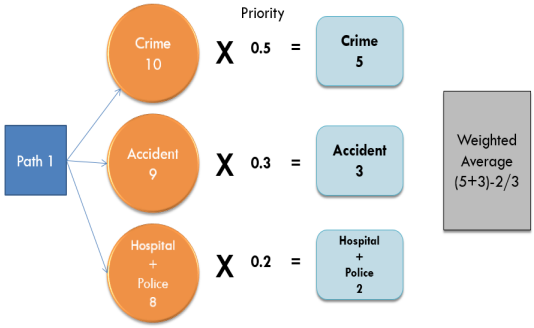

Fig. 5. Weighted Average Technique

These weightage decided in a such way that, crime events happens intentionally than accidents so giving higher priorities to those and safety services also plays major role in deciding the safeness of routes. Multiplied scores of crime and accidents are added as they contribute to unsafeness of the route and police and health services score are subtracted in weighted average calculation and the result is divided by the number of factors which is 3 gives the final result as a safe score of a single path. This way every path is given as a input to weightage average algorithm and safe score of every path is calculated individually which is processed in further algorithm. Web service then passes these generated safe scores and distance for each route to our custom approximation algorithm. This algorithm considers both safety and shortness factor and makes a best possible sequence of routes for travel. Here challenge was to balance out both factors and make best choices. Details of this algorithm as follows:

2) Approximation Algorithm: Since both the length and the risk of the path are equally important but cannot be combined into a single objective, the application's approach the urban-navigation problem as a biobjective shortest path problem and results in recommending a path that provide trade-offs between distance and safety. This algorithms is responsible for finding the trade of between two different factors as short and safe route. The outcome of this algorithm helps to suggest a path to the user among different paths suggested by Google API. The mathematical definition of approximation algorithms is the algorithms used to find approximate solutions to optimization problems and these are usually considered as NP-hard problems i.e. these problem have several different efficient solutions.

For Safe Route Recommendation, to find the approximation between safe and short, we want the one provable solution that will provide the convincing quality result. While developing the algorithm, we came across three different cases as follows:

1) The generated safe score of a path is much less but the distance of a path is much more that two paths recommended by Google service
2) The generated safe score of a path is higher i.e. path is more unsafe but the distance is very short compared to other paths
3) The resulted safe score of a path and distance both are moderate compared to other paths between same source and destination

So to build a solution which will provide both the short and safe journey, we came up with our own algorithmic approach. As the main aim of this project is to recommend the safest route, we gave the high priority to safe score and then considered the short distance whenever possible. Hence, going forward with this approach, whenever we came across the case 2 mentioned above, we eliminated the same choice in first iteration cause it completely contradicts the main goal of project. We never want to recommend the unsafe path to user even if he/she needs to compromise in the distance of a journey. So at situation we only left with two cases i.e. 1st and 3rd mentioned above. We came up with solution that if the safe score of two paths are of ratio 2:1, then will consider the path which is more safer among them for eg. if safe score of route A is 20 and safe score of route B is 43 then as the safe score difference is more than the double hence we will recommend the route A to the user even if there are noticeable differences in their distances.

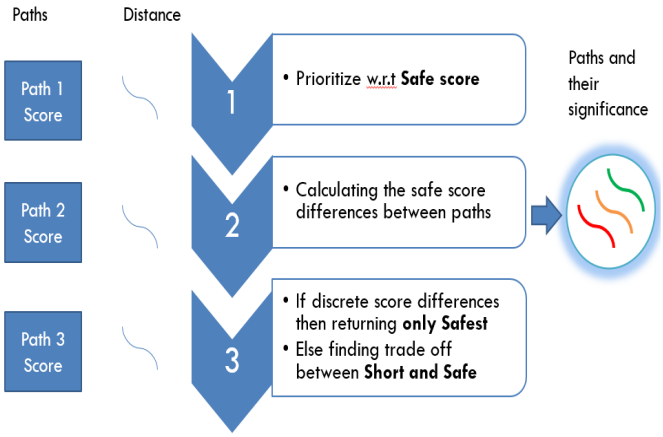

Fig. 6. Approximation Algorithm Technique

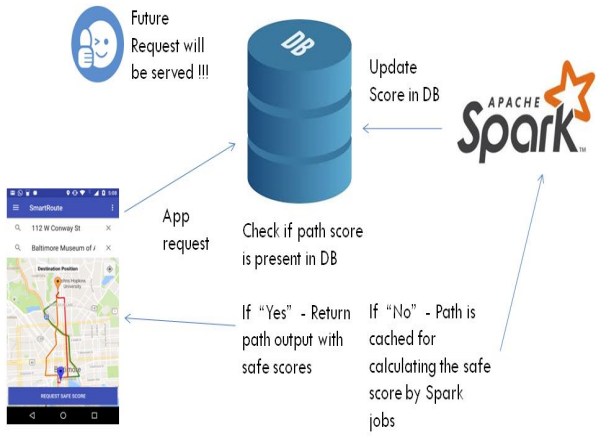

Fig. 7. Caching Approach

On the other hand if safe score difference between paths is less than the double, then will recommend the path which is more shorter and take less time for travel.

Execution of approximation algorithm generates a sequence of routes such that safest and shortest will be first, then second and so on. Finally, our web service makes JSON array of these polyline sequences and returns it to android application.

D. Path Caching

As mentioned in point B, we fetch crime, accident and healthcare services count from MongoDB. But generation of these counts involves big data processing of crime and accident data and caching mechanism. Our application works in real time and needs to generate output in few seconds. Upon request from user, it is impossible to generate counts for crime and accident by iterating over big data. Moreover, it is also impossible to preload crime and accident counts for every possible source and destination locations in world. So, we have decided to use MongoDB which serves all application queries in real time and caches paths for which counts are missing. We call table as Path Cache. When periodic spark job runs, it fetches entries from Path Cache table and generates count for each path by iterating over crime and accident data. Then this job updates the MongoDB Paths table with updated counts. It also deletes the entries in path cache table.

If crime count or accident count is not present, then we show routes based on shortest distance only. After spark job execution, all future requests for that source and destination will be served from MongoDB with updated crime and accident counts. In this way, this system will generate more scores as it receives more requests.

E. Count Generation

Above points mentioned that we are generating safe score from crime, accident and healthcare services count. These counts are generated by iterating over big data. Below point will explain this process in detail. Currently, we have dumped crime, accident and healthcare services data in HDFS. This data is well structured and stored in CSV file format. Additionally, we have designed three spark jobs for each source of data which are responsible for counting the total number of crimes, accidents or services on that path.

As explained in last point, spark job iterates over path cache collection to read polyline of every route. Then our spark job coverts these polylines into actual latitudes and longitudes. We have used polyline python library which decodes these polylines to convert them to actual co-ordinates. Our data files have latitude and longitude for every crime and accident incidence. So, our job then loads these series of latitude and longitude from data files in Spark RDD object. Then job iterates over each entry in RDD object and tries to match with input route coordinates. Here, the real challenge was to

figure out the factor for matching. We cannot simply search the exact co-ordinates of crime records with route co-ordinates.

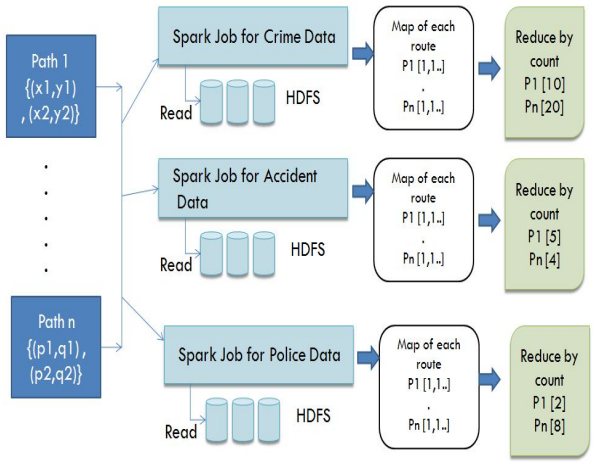

Fig. 8. Hadoop Architecture

Because practically, possibility of crime happening near the road is greater than happening on actual road. Therefore, exact match would not have worked. So, we decided that we will search around 200 meters of given path. So, for searching within 200 meters, we calculated the difference between crime location and current path coordinate. For calculating this difference, we have used geopy python library. This library calculates the difference between two methods using great circle method. The great-circle distance is the shortest distance between two points on the surface of a sphere, measured along the surface of the sphere (as opposed to a straight line through the sphere's interior). Once we have the distance, we compare with 200 meters and if its less than that then we count that record as crime happened on that path. In this way, we calculate crime and accident counts for every path and updates them to MongoDB upon finishing calculation.

X. Source code and Playstore Links

We have used GitHub to host our source code. This source code is currently availably publicly. Following are the links to source code:

https://github.com/Pratikgit/SafeRoute_backend

https://github.com/Pratikgit/SafeRouteApp

We have also deployed our android application on Google Play Store. Here is the link for downloading the application.

https://play.google.com/store/apps/details?id=edu.umbc.smartroute&hl=en

XI. Result and Analysis

In this section, we present an experimental assessment of our proposed methods. Our main objective is to evaluate the practical utility of our algorithms. We focus on two aspects: (i) the representativeness of the small set of paths that our algorithms return and (ii) their Back end processing using Spark jobs. We build our evaluation benchmark by sampling co-ordinated paths of 4 localities in Baltimore for which we have collected the required data. On providing any source and destination of these areas as a input to mobile application generates the effective result which we have verified with the safe score generated at the back end system. These results shows provable path recommendation before and after applying the safe route computations. Following are the few sample results generated at back end while processing the inputed co-ordinates:

Web Service Request

http://ec2-35-161-196-226.us-west-2.compute.amazonaws.com:5000/safe_controller/index?source=Inner%20Harbour%20Baltimore&destination=4773%20Chapel%20Square

Fig. 9. Web Service Request

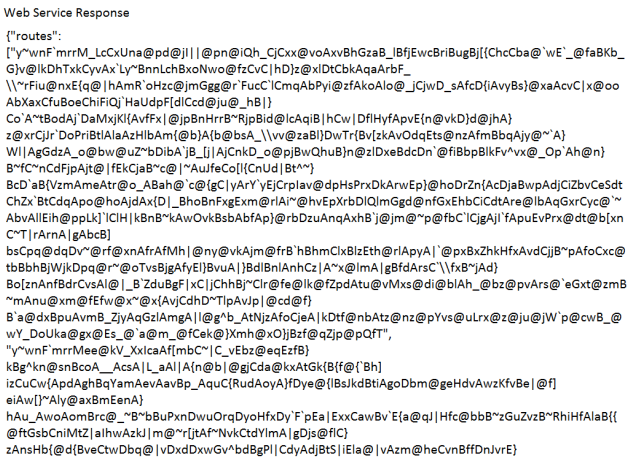

Fig. 10. Web Service Response

XII. Future Work

While experimenting we dealt with sample amount of data which is comparatively small in amount hence, we plan to scale the system and find and prepare scores for most popular cities like New York, Los Angeles etc. and feeding them into the system so that safe score will be generated for those routes. Additionally, we plan to enhance user experience, application's processing speed and build similar application for IOS users.

XIII. Conclusion

Created a mobile application that recommends different routes between source and destination according to their safety score, which is devised from algorithmic computation on authentic big data by using Hadoop Spark cluster processing. The factors deriving safety score of different routes are Criminal, Accident cases as well as number of Police and Healthcare services.

XIV. Acknowledgment

We would like to thank Dr. Milton Halem for his direction and encouragement over the course of this project and his teaching assistant Yin Huang for his lectures on Hadoop, Spark big data computation and continued support during the project.

References

[1] Esther Galbrun Konstantinos Pelechrinis Evimaria Terzi. Safe Navigation in Urban Environments
[2] Evangelos Kanoulas Yang Du Tian Xia Donghui Zhang. Finding Fastest Paths on A Road Network with Speed Patterns
[3] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. Spark: Cluster Computing with Working Sets
[4] MIC13EL X. GOEMANS, DAVID P. WILLIAMSON. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming
[5] Gerrit Sorensen Matthew Blake James Archibald Randy Beard. Safe-Path Graph Generation for Path Planning in Urban Environments
[6] Ilias Diakonikolas. Approximation of Multiobjective Optimization Problems
[7] Michael Huhns Munindar P. Singh. Service-Oriented Computing: Key Concepts and Principles
[8] https://data.cityofnewyork.us/Public-Safety/NYPD-Motor-Vehicle-Collisions/h9gi-nx95/data
[9] Daniele Quercia, Rossano Schifanella The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City
[10] http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/SingleCluster.html
[11] https://spark.apache.org/docs/0.9.1/python-programming-guide.html
[12] http://www.worldatlas.com/articles/most-dangerous-cities-in-the-united-states.html
[13] https://www.neighborhoodscout.com/neighborhoods/crime-rates/top100dangerous/
[14] http://janmatuschek.de/LatitudeLongitudeBoundingCoordinates
[15] https://developer.android.com/index.html
[16] https://developers.google.com/maps/documentation/directions/
[17] http://www.movable-type.co.uk/scripts/latlong-db.html
[18] http://dfw.cbslocal.com/2012/01/17/app-that-would-guide-users-away-from-high-crime-areas-proves-controversial/
[19] http://www.wionews.com/science-tech/five-quirky-start-up-ideas-from-techcrunch-disrupt-that-you-should-know-about-6319