AMapReduce Approach for Traffic Telemetric Analysis in Hadoop

Harshit Sharan
IDD (B.Tech+M.Tech)
Dept. of Computer Engg.
Indian Institute of Technology (BHU)
Varanasi, India

Abhinay Agrawal
Integrated Masters' Degree
Dept. of Mathematical Sciences
Indian Institute of Technology (BHU)
Varanasi, India

Abstract— **The current software systems which utilize the road traffic data of a city usually obtain it from the readily available APIs of Google Maps, Bing Maps, etc. These APIs provide the distance and time data based on the consideration that the roads are perfect and the traffic flows smoothly. However, especially in the case of countries like India, and other similar places, the traffic is not too well structured. The time and distance cannot be predicted unless a practical estimate is obtained. This paper proposes a scalable approach to analyze practically available large traffic data to provide the near perfect estimate of the traffic demography of a city. It is implemented in the open source platform of Apache Hadoop.**

Keywords—Traffic, Telemetric Analysis, Hadoop, Map Reduce, Apache, Scalability

I. INTRODUCTION

Telemetry is the science of automated communications process by which measurements are made and data is transmitted from remote sources to receiving equipment for monitoring.[1] The word is of Greek origin: tele = remote and metron = measure. Systems that need external instructions and data to work requires telecommand, the counterpart of telemetry.

Although the term commonly refers to the wireless data transfer mechanisms, it also covers data transferred over other media such as a computer network or telephone, optical link or other wired communication. Many advanced telemetry systems take advantage of the low cost and omnipresence of GSM networks by using SMS to receive and send telemetry data.

A telemeter is used to measure any quantity remotely. It consists of a sensor, a transmission path and a display, recording or control device. Telemeters are devices used in telemetry. Electronic devices are used widely in telemetry and can be wireless or hardwired, analog or digital. Other technologies are also potential, such as mechanical, optical and hydraulic.

The applications of telemetry lie diversely in the fields of meteorology, oil and gas industry, space science, motor racing, city traffic, agriculture, water management, defence, space and resource exploration, rocketry, flight testing, military intelligence, energy monitoring, resource distribution, medicine/healthcare, fishery and wildlife research and management, retail, law enforcement, energy providers, falconry, communications, etc.

In the current world scenario, devices utilize the portability and low memory requirements of operating systems like Android to collect the telemetric data.

Specifically for the software systems working on the basis of road traffic data, the current standard is of using the resources available from Application Programmer Interfaces (APIs) of Google maps, Bing maps, etc.

This data availability from the 3rd party APIs has certain limitations. The data available is appropriate for metropolitan cities and countries where the traffic flow is well ordered. For the sake of such software systems to work and be useful in case of cities in countries like India, the current running traffic analysis is required. This is because of the irregularities in the traffic demography across different roads of the city. The time required to go from one place to another, irrespective of the distances between the places, depends heavily on the current infrastructure of roads.

Thus, a system needs to be developed that analyses the traffic data and based on this analysis the actual time and distance of travel can be obtained.

The traffic data has to be collected for a long period of time. This accumulated data, that needs to be analyzed, is very large and hence the analysis system that needs to be developed should work in a distributed manner, utilizing the computing power of multiple machines.

The Apache™ Hadoop[2] project develops open source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is an open source framework for distributed storing and processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single server to thousands of machines, each providing local computation and storage. Rather than trusting on hardware to deliver availableness, the library itself is designed to detect and handle failures at the application layer, so delivering a highlyavailable service on top of a cluster of computers, each of which may be prone to failures.

MapReduce[3] is a programming model and an associated implementation for processing and generating large data sets. A map function is specified that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are representable in this model.

Programs written in this functional style are automatically parallelized and executed on a large cluster of computers. The run-time system takes care of the details of partitioning the input data, scheduling the execution of program across a set of machines, dealing machine failures, and managing the required inter-machine communication. This allows non-experienced programmers in parallel and distributed systems to easily utilize the resources of a large distributed system.

II. ALGORITHM

A. Data Collection

The traffic data can be obtained through GPS enabled devices installed on moving vehicles [4]. The devices are polled from the server periodically, and the current coordinates and other details are returned by the device to the server.

This data can be collected for a number of vehicles, over a long period of time.

B. Map Reduce Phases

The full analysis algorithm is divided in 3 separate map reduce phases. These phases are:

1. Input Collection
2. Intermediate Processing
3. Final Analysis

C. Phase 1: Input Collection

The data available from the remote device is in the form of milestone files which contain the GPS coordinates and other information pertaining to the location of the vehicle. This data is in JavaScript Object Notation (JSON)[5] format. This data can be directly converted into JAVA objects using Google's GSON[6].

The first mapred phase scans these files and gives as output the distance and time data between relevant spots. The spots are those which are already present in a database. If the vehicle is in a certain area within the vicinity of the spot, then vehicle is said to be at that spot.

The Input is given in form of a CSV file where each line corresponds to a single Duty of a vehicle. It consists of the serial numbers of the milestone files. The job can thus be equally divided by taking the modulus of the duty number by appropriate value, and treating it as the key to reducer.

The output is in the form of path files, which contains the time and distance data between various pre-defined spots available from the database.

D. Phase 2: Intermediate Processing

All the runs of the vehicles are now analyzed and the data available is in the form of Path files. Each line of the file contains the location IDs of the 2 locations between which the path is considered. It also contains the start time, end time, distance and timezone of the path. These files are in the JSON format and using the GSON API, it's directly parsed into objects. The mappers scan the files and the key of its output to the reducer is the location ID of the starting location of the path. Reducer then writes the path data in database tables, using batch queries.

The path data is saved in the database and thus is directly available using the location IDs. This step can thus be skipped if the data need not be further used, and availability from files is sufficient.

E. Phase 3: Final Analysis

This is the final stage wherein heuristics are applied to the result obtained from the previous stage. The time taken between 2 localities greatly varies according to the time of travel. Thus, separate records are stored in the database depending upon the time interval of the day in which vehicle is running. The different timezones used are:

TimeZone 1 – 0000 to 0800 hours

TimeZone 2 – 0801 to 1000 hours

TimeZone 3 – 1001 to 1200 hours

TimeZone 4 – 1201 to 1600 hours

TimeZone 5 – 1601 to 1800 hours

TimeZone 6 – 1801 to 2200 hours

TimeZone 7 – 2201 to 2400 hours

Each timezone has near similar flow of traffic on the roads. This timezone division can be modified as per the significance.

For taking the distance between any 2 localities, the minimum distance is most suitable. Also, an average of the smaller distances can be taken.

For taking the time between any 2 localities, the average time can be taken out, after removing some outliers. This is because the least time may not be the one that is most suitable.

III. EXPERIMENT AND RESULTS

The proposed algorithm was tested on a traffic data available for a period of 6 months of around 200 vehicles running in the India city of Hyderabad.

The final distance and time data was saved in the database, and tested against an already running algorithm. This algorithm previously used the distance and time between the localities directly from the Google maps. Now, the practically obtained data was used.

The data was tested on a setup which was used by a company which provides employee vehicle routing solutions to certain other firms in the city of Hyderabad. The routing engine of the company tries to minimize the distance, time taken, and waiting time of the employees while they request for drop/pickup or some random travel.

The results of this test are compiled in the below table. It has the data obtained for some random testing days on which the vehicles ran.

Using data obtained after running above algorithm												
Interval	Max Wait	Persons	Routes	Distance	Travel Time	Wait: 0-4	Wait: 5-9	Wait: 10-14	Wait: 15-19	Wait: 20-24	Wait: 25-29	Wait: 30-34
15	30	494	187	2836.09	7176.68	348	-	-	66	-	-	80
10	30	494	186	2830.10	7167.06	216	-	66	-	33	-	79
5	30	494	188	2933.88	7470.57	272	55	33	24	20	31	59
1	30	494	189	3062.24	7662.29	286	43	32	26	16	35	56
15	15	494	201	3015.52	7665.90	346	-	-	148	-	-	-
5	15	494	210	3157.72	8097.01	290	52	30	122	-	-	-
1	15	494	208	3305.90	8381.72	318	41	34	101	-	-	-

Using data obtained from Google Maps			
Persons	Trips	Distance	Travel Time
494	247	3946.42	10234.68

Table 1. Distance and Time benefits

Interval denotes the time after which the system is run and the persons are allocated vehicles. It is basically the granularity of time. Max wait is maximum time for which a person can wait before he is allocated a vehicle. Routes are the number of vehicles that will be running for the transit. Distance is the total distance travelled and travel time is the total time taken for the travel. Other columns denote the actual amount of time that the persons are waiting for. The unit of time is seconds in all the cases.

From the above table, it is clear that the total distance and time are minimized in all the cases. When the waiting time is minimized, the proposed routes are not the best, but still better than the ones that have been found by the map data.

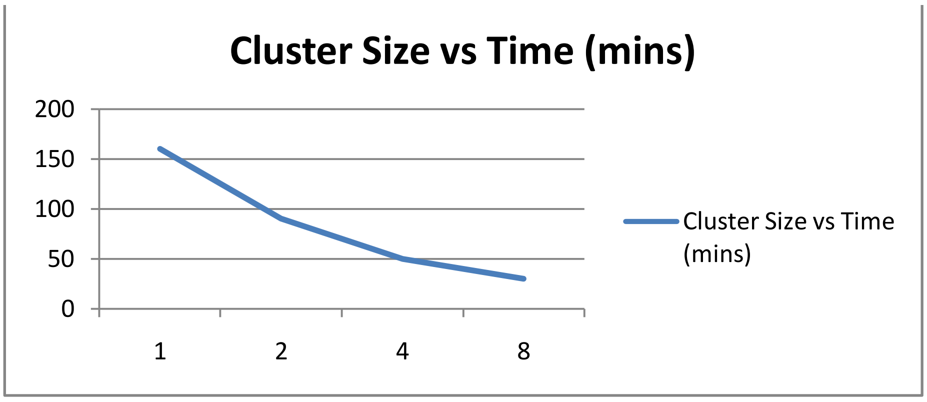

Figure 1. Cluster Size vs Time

The above chart shows the time taken by the algorithm to run on a cluster with varying number of machines.

IV. CONCLUSION AND FUTURE WORK

The major advantage of the proposed system is that it is scalable and fast. And thus it can be run periodically, and the actual changes in the traffic and demography of the roads can be regularly fed in the database.

It is seen from the chart that the time taken by the proposed algorithm reduces as the number of nodes in the hadoop cluster increases.

The total time taken, however, is mostly because of the updates made in the database. This database is present in one of the nodes in the cluster (the master node), and all other nodes transfer the data over the network to this node for update. Although batch queries are employed to transfer data in big chunks, the time would be greatly reduced if the algorithm is implemented over distributed database architecture. In future implementation, this will be incorporated.

V. REFERENCES

[1] "Telemetry: Summary of concept and rationale". NASA. NASA Technical Reports Server

[2] "Applications and organizations using Hadoop". Wiki.apache.org. 2013-06-19

[3] “MapReduce: Simplified Data Processing on Large Clusters”, Jeffrey Dean and Sanjay Ghemawat

[4] “Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment.” Juan C. Herreraa, Daniel B. Workb, Ryan Herringb, Xuegang (Jeff) Banc, Quinn Jacobsond, Alexandre M. Bayen

[5] Crockford, Douglas (May 28, 2009). "Introducing JSON"

[6] GSON Library, https://code.google.com/p/google-gson