https://dx.doi.org/10.22161/ijaers/3.11.6 *ISSN: 2349-6495(P) | 2456-1908(O)*

Accurate and Efficient Query Processing at Location-Based Services by using Route APIs

K. Bhavana[1], Dr. K. Venugopala Rao[2]

[1]M. Tech Student, Department of CSE, G. Narayanamma Institute of Technology and Science, Mandal Shaikpet, District Hyderabad, Telangana, India.

[2]Professor, Department of CSE, G. Narayanamma Institute of Technology and Science, Mandal Shaikpet, District Hyderabad, Telangana, India.

Abstract— *Efficient query processing system provides best search results to user by gathering user point of interest. Mobile users required a Location based server (LBS) to search the spatial related data. Existing system provided route results but it takes more time to execute the query and does not gives the accurate results means traffic related travel timings. The proposed system is a fastest processer for location search users. Here, LBS obtain route travel times from online route API. So it gives the accurate results to user by preventing number route request and query execution time. We use range query algorithm to reduce the number of route request and Parallel Scheduling Techniques to reduce the query execution time. Our experimental result shows that the proposed system is more efficient than existing processer.*

Keywords— Location based Server, Query Processing, Query Execution Time, Range Query, Spatial Data.

I. INTRODUCTION

Based on user point of interest, user may search the data related to street and restaurant data with location based Services. By using location based Services, a route API can find out exact travel times. Finding an optimal route in a road network between specified source and destination nodes (i.e., based on user point of interest) is one of the important things in real-world applications [10]. But whenever using route API, it takes more time to access the route. So we can reduce this problem by using Parallel Scheduling Techniques in LBS. LBS find exact results by using lower/upper bound techniques. This approach was recently shown to be very effective if lower bounds are computed using Parallel Scheduling Techniques [10]. The resulting lower bound could be used for distinguishing local and global queries for guiding local search.

Location-based server(LBS) use real-time geo-data from a mobile device or Smartphone or Computer device to provide data to user. Some services allow user to "check in" at restaurants, coffee shops, book stores, concerts, and other places or events. Often, businesses offer a reward prizes, coupons or discounts to people who check in Google Maps and Facebook Places are among the more popular services. Location-based services use a smartphone's GPS technology to find a person's location, if that person has opted-in to allow the service to do that. User can identify her area with using smartphone GPS technology without the need for manual work.

II. LITERATURE SURVEY

Web search is common in our daily lives. Caching method has been extensively used to reduce the query execution time of the search system and reduce the travel time on a road network. Another form of location related web search, known as online shortest path search, is enhanced approach due to advances in geo-positioning. However, existing caching techniques are ineffective for shortest path queries. This is due to several crucial differences between web search results and shortest path results, in relation to query matching, cache item overlapping, and query execution time. So we can manage those things by using Parallelized route requests.

If we are using LBS, It must be satisfied two things a) exact query result (i.e., exact travel timing, exact directions and exact map between sources to destination) b) less query execution time.

In Existing System **SMashQ** (Spatial mash up framework for k-NN queries) is used for best results. The k-nearest-neighbor (k-NN) query is one of the well known spatial query types for location-based services (LBS). It focus on k-NN queries in time-based road networks, where the travel time between source and destination locations may changes significantly at different time of the day [3]. In practice, it is difficult for a LBS provider to gather travel timing to find the correct route for a user to a spatial object of interest. So, we design SMashQ, a server-side spatial mashup framework that enables a database server to efficiently evaluate k-NN queries using the route data and travel time accessed from an external Web mapping service, e.g., Microsoft Bing Maps, Google API Direction. Because of the expensive cost and limitations of accessing such unnecessary external information, we propose three shared execution optimizations for

https://dx.doi.org/10.22161/ijaers/3.11.6 *ISSN: 2349-6495(P) | 2456-1908(O)*

SMashQ, to reduce the number of external route requests and provide highly accurate query answers. But it is not possible with SMashQ [13]. So here we are using range query search for accurate travel timings and less query execution time.

Use the Routes API to create a route and map that includes two or more locations and to create routes from major roads for security purpose. We can create driving or walking mode routes. Route data consists of a graphical representation of the route, a detailed turn-by-turn route description, travel times and exact directions. It enables mapping applications to provide the geographical representation of the route together with the map data, so that the route is displayed on the map to user. The Routing API is customizable so that the route calculation and additional route data can be adapted to both consumer and enterprise applications and specific application use cases. Here Routing API calculates routes between two or more points based on user interest [1] and it provides to the LBS and finally LBS send the additional route-related information like exact Direction and Travel timings to users.

III. SYSTEM DESIGN

The proposed query processing system is client-server architecture and it uses a Google route API with clients as mobile device (or) computer device and server on a computer device. This system is used to reduce query response time and number of route requests. The architecture need to meet the necessary conditions for implementing the whole system.

- First, User sends the point of interest to Location based services and this LBS integrated with route API [1].
- Second, This LBS uses the range search for accurate lower/upper bound travel time.
- Finally, It uses the parallelize route request to reduce the query response time.

System flow of the query processing can be identified as shown in Fig.1.The process initiates with a client issues a user query request. Each user having different point of interest so user search the hotels based on interest. First user send query to the location based services then these LBS's return the results with the help of Google route API. In LBS a range search algorithm is used for accurate query results (i.e., exact travel timings and directions) and less query execution time. A user sends area to server for Search the hotel. Based on area it showing the number of best hotels, map button showing the route map, travel timings and directions to the hotel, after selecting the hotel.

$$cw^{-}(e)=dist(e)/V_{MAX} \quad (1)$$

$$p.t_c^{-}=spt_{cw}-(q,p) \quad (2)$$

In second step, LBS find out the lower/upper bound travel timings with using above equations (1), (2).Where $cw^{-}(e)$ means lower bound travel time of a edge and V_{MAX} means maximum speed.

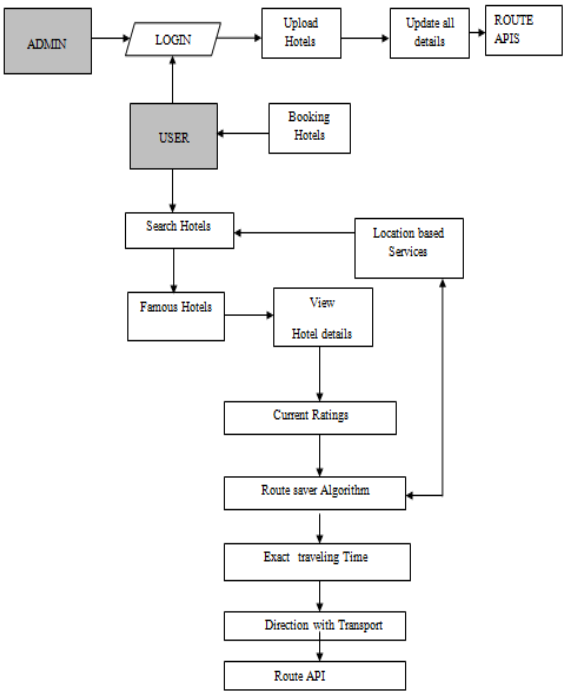

Fig.1: System flow of query processing system.

IV. FRAMEWORK

To implement the complete system the procedures section is introduced. The procedural section is examined in various sections which are:

4.1 Location based Server

This LBS takes the input as query from user and send the route request to online route API. LBS contains 3 parts and those are point of interest, route log and road network. It finds out user point of interest based on given query by using point of interest field. It sends the user interest as route request to route API. This route API provides a shortest path to LBS [8]. It stores exact routes and travel timings by using route log. Finally, it forwards the results to user.

4.2 Parallel Scheduling Techniques

Our aim is to reduce the query response time and this paper proposes advanced method for minimum query response time through parallel scheduling techniques. We can reduce the query response time by reducing number of route request. Here we use two techniques to reduce the customer response time and number of route request. Customer response time takes the more time than time spent on route request [4]. So we are considering query response time from route API's. However, Existing parallel scheduling methods having some problems related to route request because it takes extra route

https://dx.doi.org/10.22161/ijaers/3.11.6 *ISSN: 2349-6495(P) | 2456-1908(O)*

requests. Here we have two parallelization techniques to avoid the extra route requests. In first method, Greedy parallelization takes less execution time but it does not gives the exact result and another method Direction-based parallelization takes less execution time and gives the exact travel timings.

4.3 Range Query Algorithm

In this case, it presents our Route-Saver algorithm for processing a range query. It applies the travel time bounds discussed above to reduce the number of route requests. To guarantee the accuracy of returned results, it removes all expired routes in route log L. The algorithm first conducts a distance range search for P on G [6] to obtain a set C of candidate points. This algorithm consists of two phases to process the candidate points in C and store the query results in the set R. The first phase aims to shrink the candidate set C, so as to reduce the number of route requests to be issued in the second phase. First, we execute Dijkstra on G two times, using edge weight respectively.

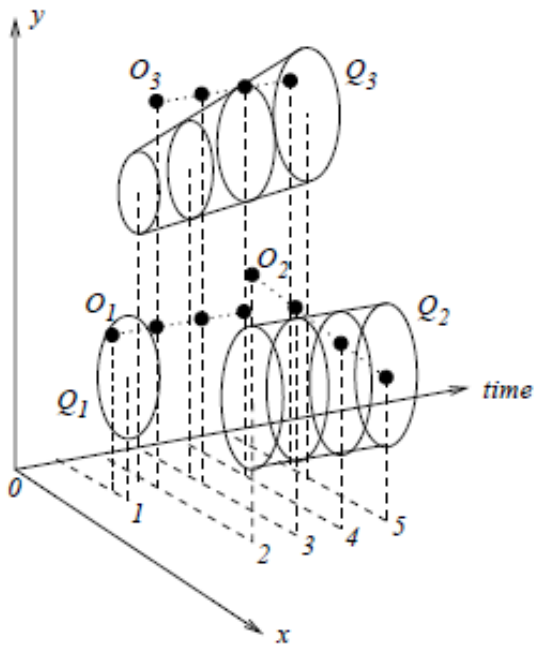

Fig.2: Predictive range query.

In predictive range query, there are three kinds of queries are distinguished based on the time span and the region they specify. Similarly, we distinguish three kinds of queries for the predictive range query: timeslice query, time-interval query and moving query. The above figure shows examples of the three kinds of predictive range queries in a 2-dimensional space. Together with the time dimension, the coordinate space is 3-dimensional. We use point objects in these examples for ease of presentation, although the following discussions also apply to objects with extents [11]. Q1 is a timeslice range query at timestamp 1. Its query region is a disk. Object O1 is in it, while O2 or O3 is not in it. Therefore the answer to Q1 is O1. *Q*2 is a time-interval range query spanning the period [2,5]. Its query region is a cylinder. At timestamp 2, no object is in *Q*2. Object *O*2 is moving and it moves into *Q*2 at timestamps 4 and 5. Objects *O*1 and *O*3 are not moving and they stay outside of *Q*2 all the time. Therefore the answer to *Q*2 is *O*2. *Q*3 is a moving range query spanning the period [2,5]. The center and radius of *Q*3 are both changing during the querying period. The query region of *Q*3 is a leaning truncated cone. No object is in *Q*3 at timestamp 2. Although *O*3 does not move, it is in *Q*3 at timestamps 4 and 5 because of the movement of *Q*3. The other two objects are outside of *Q*3 all the time. Therefore the answer to *Q*3 is *O*3. From these examples, we can see that the relative movements of objects and time are important factors to determine answers [11]. So the below route saver algorithm is used to reduce the number of route request.

Input: function Route-Saver-RANGE (Query (q,T), Data set P)

- The first phase aims to shrink the candidate set C, so as to reduce the number of route requests to be issued in the second phase.
- It execute Dijkstra on G two times so its compute the bounds p.t $_{G}^{-}$, p.t $_{G}^{+}$ and p.t $_{G}$ for every candidate p € C.
- Next, for each candidate p remaining in C, its compute exact travel time p.t $_{L}$ using optimal sub path property in L and use p.t $_{L}$ to detect true result.
- In the second phase, it issue route requests for the remaining candidates in C then insert the returned route into the route log L.
- This route provides not only the exact travel time for p, but also potential information for updating the bounds for other candidate p [1].

Output: This algorithm returns result set R to the user with accurate query result.

V. EXPERIMENTAL RESULTS

The mobile system or any system required an online route API to answer location related query and this route API integrated with location based server to provide a best search relevancy of results. However, it shows exact map between source and destination along with query execution time and here we can select the traveling mode. It shows exact travel timing based on traveling mode. Finally, it shows exact route direction to user.

https://dx.doi.org/10.22161/ijaers/3.11.6 *ISSN: 2349-6495(P) | 2456-1908(O)*

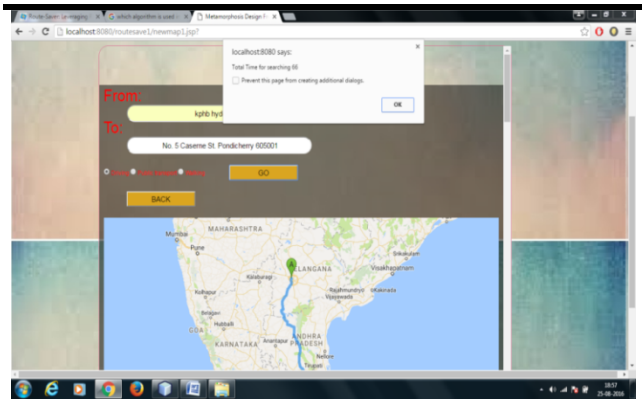

Fig.3: Showing map result

If user clicks map button in showing hotel details page then it shows the map between source and destination.

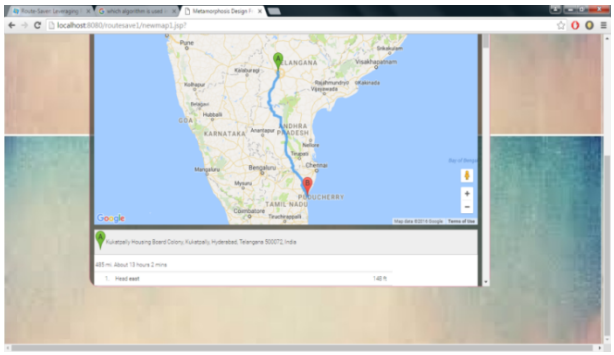

Fig.4: Showing travel times

It shows exact travel timing from user location to particular hotel.

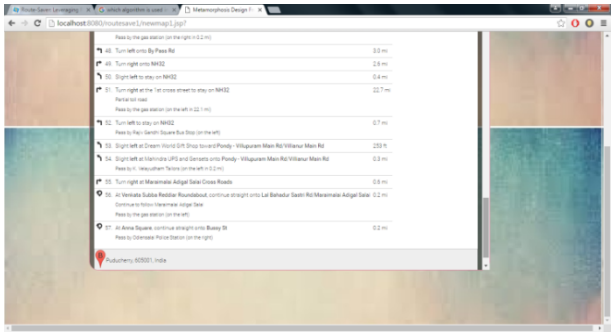

Fig.5: Showing exact directions

It shows exact direction from user location to particular hotel.

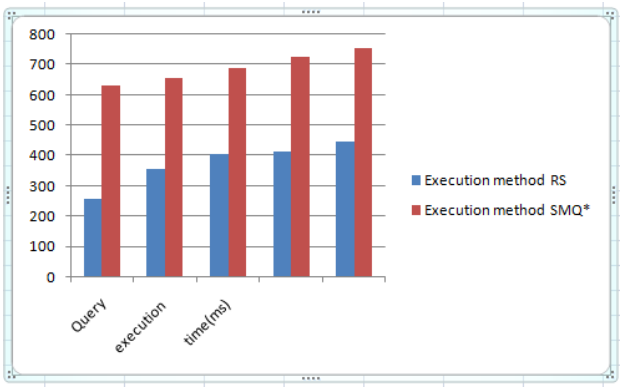

Fig.6: Showing different queries execution times

Our experiment shows the query execution time in Route saver method. We find out SMQ* query execution time based on survey. The above bar chart shows that query response time of a two execution methods and it express that route saver is more efficient than SMQ* in the point of query response time.

VI. CONCLUSIONS AND FUTURE SCOPE

It proposes a solution for the LBS to process range queries such that the query results have accurate travel times and the LBS incurs few number of web mapping requests. The time-tagged road network G and the route log L to derive lower and upper bounds of travel times for data points. During query processing, it exploits those routes to derive effective lower-upper bounds for saving web mapping requests, and examines the candidates for queries in an effective order. This solution shows that Route-Saver is more efficient than SMashQ.

In future, it can be enhanced to investigate automatic tuning the expiry time d based on a given accuracy requirement. This would help the LBS guarantee its accuracy and improve their users' satisfaction.

REFERENCES

[1] Yu Li and Man Lung Yiu, "Route-Saver: Leveraging Route APIs for Accurate and Efficient Query Processing at Location-Based Services," 2015, pp. 235-249.

[2] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in Proc. 29th Int. Conf. Very Large Data Bases, 2003, pp. 802–813.

[3] D. Zhang, C.-Y. Chow, Q. Li, X. Zhang, and Y. Xu, "SMashQ: Spatial mashup framework for k-NN queries in time-dependent road networks," Distrib. Parallel Databases, vol. 31, pp. 259–287, 2012.

[4] E. Kanoulas, Y. Du, T. Xia, and D. Zhang, "Finding fastest paths on a road network with speed patterns," in Proc. Int. Conf. Data Eng., 2006, p. 10.

[5] E. P. F. Chan and Y. Yang, "Shortest path tree computation in dynamic graphs," IEEE Trans. Comput., vol. 58, no. 4, pp. 541–557, Apr. 2009.

[6] Google Directions API. (2013). [Online]. Available:https://developers.google.com/maps/documentation/directions/

[7] H. Samet, J. Sankaranarayanan, and H. Alborzi, "Scalable network distance browsing in spatial databases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2008, pp. 43–54.

[8] J. R. Thomsen, M. L. Yiu, and C. S. Jensen, "Effective caching of shortest paths for location-based services," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2012, pp. 313–324.

[9] Pankaj K. Agarwal, "Range Searching", Available:https://www.cs.duke.edu/~pankaj/publications/surveys/.

https://dx.doi.org/10.22161/ijaers/3.11.6 *ISSN: 2349-6495(P) | 2456-1908(O)*

[10] Peter Sanders and Dominik Schultes," Robust, Almost Constant Time Shortest-Path Queries in Road Networks", Available: http://algo2.iti.kit.edu/schultes/hwy/hhTransit.pdf

[11] Rui Zhang1, H. V. Jagadish2, Bing Tian Dai3, Kotagiri Ramamohanarao4," Optimized Algorithms for Predictive Range and KNN Queries on Moving Objects", Available: http://www.ruizhang.info/publications/IS_MovingRangeKnn.pdf.

[12] U. Demiryurek, F. B. Kashani, C. Shahabi, and A. Ranganathan, "Online computation of fastest path in time-dependent spatial networks," in Proc. 12th Int. Symp. Adv. Spatial Temporal Databases, 2011, pp. 92–111.

[13] Wan D. Bae1 , Shayma Alkobaisi1 , Seon Ho Kim1 , Sada Narayanappa1, "Supporting Range Queries on Web Data Using *k*-Nearest Neighbor Search", Available: http://infolab.usc.edu/DocsDemos/w2gis.pdf .